ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 37 Выпуск 6 Часть 2

ISSN Online 2220-6426
Volume 37 Issue 6 Part 2

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2025

ИСП РАН

ISSN 2079-8156 (Print), ISSN 2220-6426 (Online)

Труды Института системного программирования РАН
Proceedings of the Institute for System Programming of the RAS

Труды ИСП РАН – это издание с двойной анонимной системой рецензирования, публикующее научные статьи, относящиеся ко всем областям системного программирования, технологий программирования и вычислительной техники. Целью издания является формирование научно-информационной среды в этих областях путем публикации высококачественных статей в открытом доступе.

Издание предназначено для исследователей, студентов и аспирантов, а также практиков. Оно охватывает широкий спектр тем, включая, в частности, следующие:

- операционные системы;
- компиляторные технологии;
- базы данных и информационные системы;
- параллельные и распределенные системы;
- автоматизированная разработка программ;
- верификация, валидация и тестирование;
- статический и динамический анализ;
- защита и обеспечение безопасности ПО;
- компьютерные алгоритмы;
- искусственный интеллект.

Журнал издается по одному тому в год, шесть выпусков в каждом томе.

Поддерживается открытый доступ к содержанию издания, обеспечивая доступность результатов исследований для общественности и поддерживая глобальный обмен знаниями.

Труды ИСП РАН реферируются и/или индексируются в:

Proceedings of ISP RAS are a double-blind peer-reviewed journal publishing scientific articles in the areas of system programming, software engineering, and computer science. The journal's goal is to develop a respected network of knowledge in the mentioned above areas by publishing high quality articles on open access.

The journal is intended for researchers, students, and practitioners. It covers a wide variety of topics including (but not limited to):

- Operating Systems.
- Compiler Technology.
- Databases and Information Systems.
- Parallel and Distributed Systems.
- Software Engineering.
- Software Modeling and Design Tools.
- Verification, Validation, and Testing.
- Static and Dynamic Analysis.
- Software Safety and Security.
- Computer Algorithms.
- Artificial Intelligence.

The journal is published one volume per year, six issues in each volume.

Open access to the journal content allows to provide public access to the research results and to support global exchange of knowledge. **Proceedings of ISP RAS** is abstracted and/or indexed in:

УДК004.45

Адрес: 109004, г. Москва, ул. А. Солженицына, дом 25.
Телефон: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Сайт: http://www.ispras.ru/proceedings/

Address: 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Tel: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Web: http://www.ispras.ru/en/proceedings

Содержание

Table of Contents

DOI: 10.15514/ISPRAS-2025-37(6)-16

Исследование эффективности планировщиков протокола MPQUIC в зависимости от алгоритмов контроля перегрузки

[1,2] *М.В. Попов, ORCID: 0009-0007-7774-2220 <mpopov@ispras.ru>*
[1,3] *И.А. Степанов, ORCID: 0009-0003-1964-5001 <ivan_mipt@ispras.ru>*
[1,2,3,4] *А.И. Гетьман, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*

[2] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, г. Москва, Ленинские горы, д. 1.*

[3] *Московский физико-технический институт, 141700, Россия, Московская область, г. Долгопрудный, Институтский пер., 9.*

[4] *Национальный исследовательский университет «Высшая школа экономики», 101978, Россия, г. Москва, ул. Мясницкая, д. 20.*

Аннотация. В последние годы широкую популярность получил протокол QUIC, как альтернатива TCP. Кроме того, в настоящее время широко внедряется и исследуется технология Multipath, реализованная в протоколе MPQUIC. Центральным компонентом протокола MPQUIC является планировщик, принимающий решение по какому пути и в какой момент времени отправить следующие пакеты данных. Существуют реализации планировщиков как на основе эвристических правил, так и на основе обучения с подкреплением. На данный момент поведение планировщиков в различных, с точки зрения характеристик путей, средах изучено подробно. Однако вопрос их эффективности в зависимости от используемых алгоритмов контроля перегрузки недостаточно освящён. В данной работе представлена реализация различных планировщиков и исследование их эффективности в зависимости от алгоритма контроля перегрузки. Полученные результаты, на основе проведённых экспериментов, говорят о том, что планировщик может эффективно работать в сетевой среде с определённым алгоритмом контроля перегрузки, но при этом быть не эффективным в среде с другим алгоритмом контроля перегрузки.

Ключевые слова: Многопутевой планировщик; протокол QUIC; протокол MPQUIC; алгоритмы контроля перегрузки; обучение с подкреплением.

Для цитирования: Попов М.В., Степанов И.А., Гетьман А.И. Исследование эффективности планировщиков протокола MPQUIC в зависимости от алгоритмов контроля перегрузки. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 7–20. DOI: 10.15514/ISPRAS–2025–37(6)–16.

Research of the Effectiveness of MPQUIC Protocol Schedulers Depending on the Congestion Control Algorithms

[1,2] *M.V. Popov, ORCID: 0009-0007-7774-2220 <mpopov@ispras.ru>*
[1,3] *I.A. Stepanov, ORCID: 0009-0003-1964-5001 <ivan_mipt@ispras.ru>*
[1,2,3,4] *A.I. Getman, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[2] *Lomonosov Moscow State University, 1 Leninskie Gory, Moscow, 119991 Russia.*
[3] *Moscow Institute of Physics and Technology (National Research University), 9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia.*
[4] *National Research University «Higher School of Economics», 20 Myasnitskaya ulitsa, Moscow 101000 Russia.*

Abstract. In recent years, the QUIC protocol has become widely popular as an alternative to TCP. In addition, Multipath technology implemented in the MPQUIC protocol is currently being widely implemented and researched. The central component of the MPQUIC protocol is the scheduler, which decides which path and at which time to send the next data packets. There are implementations of schedulers based on both heuristic rules and reinforcement learning. At the moment, the behavior of schedulers in various environments has been studied in detail in terms of path characteristics. However, the issue of their effectiveness, depending on the congestion control algorithms used, is not sufficiently sanctified. This paper presents the implementation of various schedulers and a study of their effectiveness depending on congestion control. The results obtained suggest that the scheduler can work effectively in a network environment with a certain congestion control algorithm, but it may not be effective in an environment with a different congestion control algorithm.

Keywords: Multipath scheduler; QUIC protocol; MPQUIC protocol; congestion control algorithms; reinforcement learning.

For citation: Popov M.V., Stepanov I.A., Getman A.I. Research of the effectiveness of MPQUIC protocol schedulers depending on the Congestion Control Algorithms. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 7-20 (in Russian). DOI: 10.15514/ISPRAS-2025-37(6)-16.

1. Введение

Multipath (многопутевая передача данных) — это технология, позволяющая одновременно использовать несколько сетевых интерфейсов (4G, 5G, Wi-Fi и т.д.) для передачи данных. Главная цель данной технологии заключается в повышении производительности и отказоустойчивости. Первоначально данная технология была реализована для протокола TCP в виде многопутевого протокола MPTCP [1-3].

В последнее время протокол QUIC стал альтернативой протоколу TCP и получил популярность благодаря высокой производительности. Поэтому в последствии были реализованы технологии multipath для QUIC в виде протокола MPQUIC [4-5].

Принципиальное отличие протокола QUIC от протокола MPQUIC представлено на рис. 1.

Центральным элементом протокола MPQUIC является планировщик – компонент, отвечающий за распределение данных между несколькими доступными сетевыми путями таким образом, чтобы обеспечивать высокую производительность, отказоустойчивость и минимизацию задержек. Планировщик взаимодействует с алгоритмом контроля перегрузки. Стоит отметить, что эти два компонента решают одну общую задачу – оптимизацию производительности, но используют разные методы. Планировщик решает, как распределить данные между несколькими путями, а алгоритмы контроля перегрузки (Reno, Cubic, BBR) управляют интенсивностью передачи данных на каждом пути, чтобы избежать перегрузок сети.

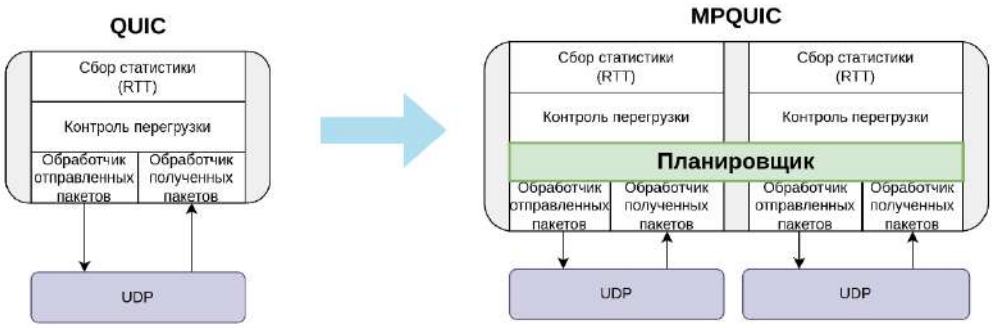


Рис. 1. протокол QUIC и протокол MPQUIC.
Fig. 1. QUIC protocol and MPQUIC protocol.

В настоящее время существующие планировщики можно разбить на два класса: основанные на эвристических правилах и основанные на обучении с подкреплением. К недостаткам первых относится неэффективная работа в динамически меняющихся средах, к недостаткам вторых - высокая вычислительная сложность и возможное переобучение. Стоит отметить, что на работу планировщика может значительно влиять алгоритм контроля перегрузки, который управляет скоростью передачи данных. Однако в известных нам работах не представлены исследования эффективности реализованных планировщиков в зависимости от алгоритмов контроля перегрузки.

В данной работе представлены реализации планировщиков как на основе обучения с подкреплением, так и на основе эвристических правил, и представлено сравнение их эффективности в зависимости от использованного алгоритма контроля перегрузки.

Остальная часть работы организована следующим образом. В разделе 2 представлена постановка задачи и даны основные определения по данной теме. В разделе 3 описаны популярные планировщики, основанные как на эвристических правилах, так и на обучении с подкреплением. Раздел 4 содержит описание планировщиков на основе обучения с подкреплением, реализованных в данной работе. В разделе 5 представлены эксперименты эффективности реализованных планировщиков в различных сетевых сценариях и при использовании различных алгоритмов контроля перегрузки. В разделе 6 подводятся итоги на основе полученных результатов и обсуждаются направления дальнейших исследований.

2. Постановка задачи

В данном разделе будут рассмотрены базовые определения, связанные с данной темой и необходимые для постановки задачи.

2.1 Определения

Задержка (delay) – время, необходимое для передачи данных от отправителя к получателю.

Время приема-передачи (Round trip time, RTT) – время, необходимое для передачи данных от отправителя к получателю и получения подтверждения, что данные были получены.

SRTT (Сглаженное RTT): $SRTT_n = \frac{7}{8} SRTT_{n-1} + \frac{1}{8} RTT_{last}$, RTT_{last} — последнее значение данного параметра.

CWND (окно перегрузки): максимальное количество данных (в байтах или сегментах), которое отправитель может передать получателю через сеть до получения подтверждения о доставке этих данных без учёта окна приёма RWND.

RWND (окно получателя): окно приёма, которое указывает, сколько данных готов принять получатель (определяется буфером получателя).

SWND (окно отправления): максимальное количество данных (в байтах), которые отправитель может передать получателю до получения подтверждений (ACK). SWND определяется как $min(CWND, RWND)$.

Bytes in Flight (P, байты в полёте): число отправленных ещё не подтверждённых байт.

Пропускная способность (throughput) – отношение объёма данных, проходящих через сеть (канал) за заданный интервал времени, к данному интервалу времени. Пропускная способность является изменяемой величиной и показывает скорость отправки данных, в то время как **ширина канала (bandwidth)** – это величина неизменная, являющаяся максимально возможной пропускной способностью.

Capacity (мера свободы канала) – отношение пропускной способности к ширине канала в данный момент времени.

Loss (потери) пакетов – число потерянных пакетов за некоторый интервал времени.

Loss_rate (скорость потери пакетов) – отношение числа потерянных пакетов за заданный интервал времени к данному интервалу времени.

2.2 Постановка задачи Multipath

Как уже отмечалось ранее, технология Multipath предназначена для повышения производительности и отказоустойчивости. Более строго постановку данной задачи можно записать следующим образом. Пусть существует N доступных путей, каждый из которых имеет следующие характеристики:

- RTT_i- RTT i-го пути
- L_i - процент потерь i-го пути
- r_i - пропускная способность i-го пути
- C_i - ширина i-го канала

Тогда задача технологии Multipath выглядит следующим образом:

$$\alpha \sum_{i=1}^{N} r_i - \beta \sum_{i=1}^{N} RTT_i - \gamma \sum_{i=1}^{N} L_i \to max \quad (1)$$
$$r_i < C_i \forall i \in [1 \dots N]$$

α, β, γ - некоторые числовые коэффициенты.

Стоит отметить, что задачи максимизации пропускной способности и минимизации RTT могут быть ортогональными. Так как для увеличения пропускной способности необходимо максимально использовать доступную ширину канала сети. Это достигается за счёт более агрессивного заполнения буферов маршрутизаторов и коммутаторов, что позволяет передавать больше данных. Это в свою очередь приводит к возникновению очередей, что увеличивает RTT. Поэтому в данной постановке задачи реализованный алгоритм должен соблюдать баланс между минимизацией RTT и максимизацией пропускной способности.

С точки зрения обучения с подкреплением агент, который является планировщиком, выбирает действие на основе некоторой политики. Действием обычно является выбор некоторого пути или ожидания освобождения окна перегрузки. После выбора действия среда генерирует новое состояние и награду. Состоянием является некоторое описание характеристики сети в данный момент. Наградой выступают метрики, значения которых оптимизируются в формуле 1.

2.3 Алгоритмы контроля перегрузки

Как уже отмечалось ранее в данной работе большое внимание уделено исследованию технологии Multipath для различных алгоритмов контроля перегрузки, таких как CUBIC, BBRv1 и BBRv2. Упомянутые алгоритмы являются наиболее популярными на данный момент, поэтому были выбраны в качестве исследований. Данные алгоритмы были впервые

реализованы для протокола TCP, однако впоследствии появились реализации для QUIC, учитывающие особенность данного протокола.

New Reno (1999 г.) [6]. Данный алгоритм является улучшенной версией классического алгоритма TCP Reno (1990 г.), который включал в себя экспоненциальное или линейное увеличение окна перегрузки в случае работы без потерь и уменьшение окна перегрузки в случае потерь. Хоть New Reno крайне стабильный алгоритм, однако, зачастую, использует ширину канала не столь эффективно, как алгоритмы, появившиеся позднее, и поэтому в данной работе он подробно не исследуется.

Cubic (2005 г.) [7]. Вместо линейного увеличения окна перегрузки, данный алгоритм использует кубическую функцию, которая зависит от времени, прошедшего с момента последней потери пакета. Таким образом CUBIC эффективен в каналах с высокой пропускной способностью и на данный момент является одним из самых популярных алгоритмов контроля перегрузки.

BBRv1 (2016 г.) [8]. В отличие от предыдущих алгоритмов, определяющих перегрузку по потери пакетов, данный алгоритм использует модель, основанную на оценке ширины канала и минимальной задержке для определения оптимального размера окна. Этот размер окна соответствует идеальному объему данных, который может быть передан по сети без перегрузки и без необходимости дожидаться потерь пакетов для корректировки скорости передачи. Таким образом BBRv1 использует ширину канала максимально эффективно, однако может не обеспечивать равномерное распределение пропускной способности.

BBRv2 (2018 г.). Вторая версия алгоритма BBR стремится к лучшей справедливости при разделении полосы пропускания с другими соединениями, устраняя таким образом недостатки первой версии. Однако BBRv2 в силу своей справедливости использует ширину канала не столь эффективно.

3. Обзор существующих решений

В данном разделе будут рассмотрены популярные планировщики, основанные как на эвристических правилах, так и на обучении с подкреплением.

3.1 Планировщики, основанные на эвристических правилах

3.1.1 Round Robin (RR). Один из самых простых планировщиков, который последовательно и циклично отправляет пакеты по доступным путям. Данный планировщик не требует сложных вычислений состояний пути, что неизменно является его достоинством. Однако стоит отметить существенные недостатки данного метода. Во-первых, в том случае, если один путь заведомо лучше других, планировщик не будет учитывать данное обстоятельство и эффективно использовать лучший путь. Во-вторых, в случае динамически меняющейся среды Round Robin не может адаптироваться к новым условиям. Так как данный планировщик не учитывает состояние сетевой среды, в которой работает, то в дальнейшем рассматриваться Round Robin не будет.

3.1.2 MinRTT. RTT является важной характеристикой пути, указывающей на задержку при передаче данных. Планировщик minRTT учитывает эту характеристику, отправляя пакет в путь с минимальным значением RTT, в том случае, если путь доступен. Это может быть важно для приложений, чувствительных к задержкам (видеоприложения, VoIP и т.д.). Данный планировщик может быстро реагировать на изменения в состоянии сети, в том случае если RTT путей увеличивается или уменьшается. Однако использование в качестве метрики эффективности только RTT может быть недостаточным в ряде случаев. К примеру, в том случае, если первый путь имеет низкий RTT и низкую пропускную способность, а второй путь имеет RTT чуть выше и очень высокую пропускную способность, алгоритм будет передавать пакеты по первому пути, что ограничивает общую производительность, так как второй более эффективный путь не используется. Вторая проблема планировщика

minRTT при выборе пути с минимальным RTT заключается в том, что если в какой-то момент быстрый путь становится недоступным, то пакеты отправляются в медленный путь. Поэтому может возникнуть ситуация, при которой быстрый путь будет свободен, однако пакеты были отправлены по медленному пути.

3.1.3 BLEST. Планировщик BLEST [9] отчасти решает проблему планировщика minRTT следующим образом: в том случае, если быстрый путь недоступен, BLEST оценивает будущее заполнение окон перегрузки на уровне соединения. В том случае, если оценка покажет, что пути будут заполнены, планировщик ожидает, а не отправляет пакеты в более медленный второй путь. Стоит отметить, что в своих оценках BLEST помимо RTT использует также CWND и число байт в полёте для оценки эффективности пути.

3.1.4 ECF. Данный планировщик [10] выбирает путь с минимальным RTT. В том случае, если данный путь не доступен, то ECF выбирает путь с минимальным ожидаемым временем доставки. Оценка ожидаемого времени доставки также основана на информации об RTT и CWND, что отличает данный планировщик от minRTT с точки зрения оценки эффективности путей.

3.2 Планировщики, основанные на обучении с подкреплением

Главным недостатком планировщиков, основанных на эвристических правилах, является их низкая адаптируемость к динамически меняющимся условиями среды в силу того, что они работают на основе определённых правил. В данном подразделе будут рассмотрены популярные планировщики на основе RL и выделены их особенности с точки зрения задачи обучения с подкреплением.

3.2.1 Peekaboo. Данный планировщик [11] основан на алгоритме Linear Upper Confidence Bound, являющимся улучшением классического алгоритма UCB. В качестве состояний авторами выбраны нормализованные значения отношений CWND, SWND и числа байт в полёте (P) к RTT для каждого из двух путей. Таким образом, вектор состояний имеет 6 элементов. Награда представляет собой сумму наград за каждое действие с учётом коэффициента дисконтирования. Награда за каждое действие (отправки пакета) определяется как отношение между размером пакета в байтах и временем, прошедшим с момента передачи пакета до получения подтверждения (ACK). Набор действий зависит от доступности путей. Авторы рассматривают два случая. Если доступен только один путь, набор действий включает в себя: передачу по этому пути или ожидание до тех пор, пока другой путь снова не станет доступным; если доступны оба пути, то набор действий включает в себя: передачу пакета по первому пути или же передачу пакета по второму пути. Авторами был проведён анализ работы алгоритма в различных сетевых средах с точки зрения ширины канала, задержки, дисперсии RTT и уровня потерь по сравнению с классическими планировщиками, такими как: RR, minRTT, BLEST и ECF.

3.2.2 M-Peekaboo. Данный планировщик [12] был представлен как расширение планировщика Peekaboo для сетей 5G. С точки зрения обучения с подкреплением M-Peekaboo отличался от своего предшественника незначительно. Главным вкладом авторов было исследование планировщика в сценарии сетей 5G, то есть сетей со следующими параметрами: ширина канала (Bandwidth) = 1100Mbps, RTT = 27.4 ± 6.4 ms, доля потерь пакетов = 0.01.

3.2.3 Планировщик MPQUIC на основе DQN. Данный планировщик [13] основан на глубоком Q-обучении и был исследован в сценарии потоковой передачи видео. Одним из особенностей данного планировщика является принятие решений не для каждого отдельного пакета, а для всех в течении определённого интервала времени (50ms). Это сделано по двум причинам: во-первых, частое принятие решений требует высоких вычислительных затрат, во-вторых, из-за задержки в сети требуется время, чтобы узнать, был ли успешно передан запланированный пакет, то есть было ли успешным то или иное действие. Авторы расширили

пространство состояний, включив помимо классических RTT, CWND, также SRTT, размер окна отправки (SWND) и т.д. В качестве действий выбрано случайное действие с некоторой вероятностью или же выбор пути с минимальным RTT. Реализованный планировщик исследовался в следующих сетевых сценариях: сеть без потерь и с разнообразным RTT для путей, сеть с потерями и одинаковыми путями с точки зрения RTT.

3.2.4 Планировщик MPQUIC на основе DQN для 5G. Данный планировщик [14] был предложен для сетей 5G и использовал классические состояния: SRTT, CWND и число байт в полёте. Итоговая награда определялась как средняя пропускная способность в мегабитах в секунду. Кроме того, в качестве вознаграждения был предусмотрен штраф при достижении некоторого количества пакетов без подтверждения или повторных пакетов. В качестве действий авторами было выбрана отправка в лучший путь (с точки зрения RTT) или ожидание. Основой планировщика является нейронная сеть, представленная классической архитектурой для задачи DQN: небольшое число скрытых слоёв с функцией активацией ReLU. К сожалению, эксперименты были проведены в узком диапазоне сетевых характеристик и не в реальных сетевых условиях, но в тоже время, в данной работе было представлено сравнение эффективности передачи данных при использовании технологии Multipath с передачей данных без использования данной технологии.

3.2.5 FALCON. Ещё один планировщик [15], разработанный для сценария 5G. Данный планировщик основан на DQN и является продолжение планировщика M-Peekaboo, однако главным его отличием является парадигма адаптированного обучения. Использование адаптированного обучения повышает эффективность планировщика в различных сетевых средах, рассматриваемых авторами: в средах с высокой пропускной способностью, низкой пропускной способностью и так далее. Сетевые среды исследовались с точки зрения характеристик сетевых каналов, однако исследования сред с точки зрения алгоритмов контроля перегрузки выполнено не было. Сравнение планировщиков на основе обучения с подкреплением представлено в табл. 1.

4. Реализованные планировщики на основе обучения с подкреплением

Помимо классических планировщиков minRTT, ECF и BLEST в данной работе были реализованы планировщики на основе Q-learning, DQN и LinUCB. В данном разделе подробно описаны архитектуры последних трёх.

4.1 Q-learning

Планировщик, основанный на Q-обучении, вначале своей работы определяет путь с минимальным RTT (*path_f*). Затем планировщик определяет путь с минимальным RTT, который является доступным (*path_s*). Доступность определяется заполненностью окна перегрузки у пути. В том случае, если *path_f* не равен *path_s* запускается алгоритм Q-обучения.

Состояния: для каждого из двух активных путей вычисляется класс производительности (низкий, средний, высокий), полученный из оценки качества пути.

- **Низкий класс** – высокий RTT или большое количество срабатываний PTO или высокие потери.
- **Средний класс** – средний RTT, низкая пропускная способность пути или умеренные потери.
- **Высокий класс** – RTT низкий, потери малы, PTO не срабатывает.

Так как каждый путь может иметь 3 класса, а пути всего два, то, следовательно, среда имеет всего 9 состояний.

Действия: действие1 – отправить пакет по первому пути, действие2 — отправить пакет по второму пути.

Награда: $\alpha(throughtput_1 + throughtput_2) - \beta(RTT_1 + RTT_2)$.

При этом значения каждой из 4 переменных были нормализованы.

Табл. 1. Сравнение различных алгоритмов Multipath.
Table 1. Comparing different learning algorithms Multipath.

N	Год	Состояния	Награды	Действия
11	2020	CWND, SWND, lnP, RTT	throughtput	1. ожидание свободного пути 2. передача по быстрому пути
12	2021	CWND, RTT, lnP	throughtput	1. ожидание свободного пути 2. передача по быстрому пути
13	2022	CWND, SRTT, P и ещё 7 состояний	подтверждённые пакеты	1. ожидание свободного пути 2. передача по быстрому пути
14	2019	CWND, SRTT, P и ещё 3 состояния	throughput	1. отправка пакета в путь 1 2. отправка пакета в путь 2
15	2022	CWND, SWND, RTT, P	throughput	1. отправка пакета в путь 1 2. отправка пакета в путь 2

4.2 Deep Q-learning

В данном планировщике табличное представление функции качества Q(s,a), применявшееся в базовом Q-learning, заменено аппроксимацией на основе полносвязной нейронной сети. Полносвязная нейронная сеть имеет 6 входных признаков, два скрытых слоя по шесть нейронов и два выходных нейрона. В качестве функции активации использовалась ReLU.

Планировщик, основанный на глубоком Q-обучении, вначале своей работы определяет путь с минимальным RTT (*path_f*). Затем планировщик определяет путь с минимальным RTT, который является доступным (*path_s*). Доступность определяется заполненностью окна перегрузки у пути. В том случае, если *path_f* не равен *path_s* запускается алгоритм глубокого Q-обучения.

Состояния: $CWND_1$, $CWND_2$, $SRTT_1$, $SRTT_2$, Bytes in $Flight_1$, Bytes in $Flight_2$.

Действия: действие1 – ожидать освобождения первого (наиболее быстрого) пути, действие2 – отправить пакет по второму пути.

Награда: $\alpha(throughtput_1 + throughtput_2) - \beta(RTT_1 + RTT_2)$ (компоненты награды нормализованы).

4.3 Linear Upper Confidence Bound (UCB)

Основная идея данного планировщика была взята из [11]. Алгоритм принимает решение, по какому из двух активных путей отправить пакет, рассматривая это как задачу Linear Upper

Confidence Bound (LinUCB) с двумя действиями. Планировщик, основанный на LinUCB, вначале своей работы определяет путь с минимальным RTT (*путь F*) и путь с большим RTT, чем у пути F, но меньшим, чем у всех остальных (*путь S*). Если у *пути F* ещё есть свободное окно CWND, пакет сразу отправляется по нему, иначе запускается алгоритм обучения LinUCB.

Состояния x: для каждого из *путей F и S* рассчитывается отношение текущего окна CWND к усреднённому RTT, отношение RWND к усреднённому RTT и количество байт в полёте к усреднённому RTT соответствующего пути.

Действия: действие1 – отправить пакет по *пути F*, действие2 – отправить пакет по *пути S*. Для каждого действия хранится матрица A размера 6 × 6 и вектор b длиной 6 – вектор накопленных наград.

Награда: $\alpha(throughtput_1 + throughtput_2) - \beta(RTT_1 + RTT_2)$.

Основные этапы работы планировщика заключаются в следующем:

1. Для текущего вектора состояний x вычисляется прогноз эффективности каждого действия: $val_F = (A_F^{-1}b_F)^T x$ и $val_S = (A_S^{-1}b_S)^T x$.
2. В том случае, если $val_S < val_F$, то со случайной вероятностью 0.3 выбирается *путь S*, а в другом случае происходит ожидание *пути F*.
3. В том случае, если $val_S \geq val_F$, то со случайной вероятностью 0.9 выбирается *путь S*, а в другом случае происходит ожидание *пути F*.
4. Рассчитывается накопленная награда на основе предыдущей формулы.
5. Расчёт $theta_F = A_F^{-1} \cdot b_F$ и $theta_F = A_S^{-1} \cdot b_S$ следующим образом: $\theta[i] = \theta[i] + lr \cdot (reward - val)$.

Таким образом, данный планировщик **реализует** идеи эвристических методов, используя вероятностный выбор, и обучения с подкреплением, используя Linear Upper Confidence Bound (LinUCB).

5. Результаты

В данном разделе содержатся основные результаты, полученные при проведении экспериментов.

5.1 Экспериментальный стенд

Для проведения исследований использовалась среда Mininet [16], с помощью которой была создана топология, состоящая из MPQUIC-клиента и MPQUIC-сервера, каждый из которых имел два сетевых интерфейса. Эти интерфейсы были подключены к двум независимым маршрутам передачи данных, каждый из которых мог иметь различные характеристики. Топология сети представлена на рис. 2.

В ходе эксперимента клиент отправлял серверу файлы различного размера, используя протокол QUIC. В качестве конкретной реализации протокола QUIC была взята реализация XQUIC [17]. Стоит отметить, что в XQUIC присутствует всего один планировщик – minRTT. Поэтому в данной работе были реализованы также следующие планировщики: ECF, BLEST, основанный на Q-обучении, основанный на DQN, основанный на LinUCB. Для каждого сценария и каждого алгоритма контроля перегрузки было выполнено 10 запусков и замеров времени передачи файлов.

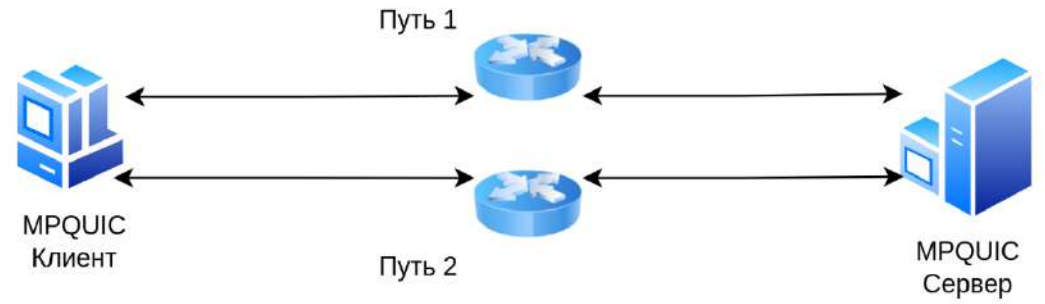


Рис. 2. Топология сети.
Fig. 2. Network topology.

5.2 Характеристики экспериментов

Наиболее важными с точки зрения характеристик путей являются: ширина канала, задержка, процент потерь и дисперсия задержки. Характеристики путей в различных экспериментах представлены в табл. 2.

Табл. 2. Характеристики путей.
Table 2. Paths characteristics.

N	Ширина каналов (Mb/s)		Задержка (мс)		Процент потерь (%)		Дисперсия задержки ($мс^2$)	
	Путь №1	Путь №2	Путь №1	Путь №2	Путь №1	Путь №2	Путь №1	Путь №2
1	70	20	40	10	0.5	1	5	5
2	70	20	10	40	0.5	1	5	5
3	70	70	15	15	0	0	0	0

Сценарий № 1 соответствует ситуации, при которой первый путь имеет высокую ширину канала и высокую задержку относительно второго пути. При этом, каждый путь имеет определенную дисперсию задержки и число потерь. Данный сценарий моделирует 4G или Wi-Fi соединение, при котором наблюдается нестабильное соединение. Также стоит отметить, что в таком случае сложно определить какой из двух путей является лучшим с точки зрения эффективности.

Сценарий № 2 соответствует ситуации, при которой первый путь имеет высокую ширину канала и низкую задержку относительно второго пути. При этом, каждый путь имеет определенную дисперсию задержки и число потерь. Данный сценарий также моделирует 4G или Wi-Fi соединение, при котором наблюдается нестабильное соединение. Главное отличие заключается в том, что в таком случае существует путь (путь 1), который является предпочтительнее с точки зрения эффективности.

Сценарий № 3 соответствует ситуации, при которой оба пути имеют одинаковые характеристики. При этом каждый путь не имеет дисперсию задержек и потерь. Данный сценарий моделирует 4G или Wi-Fi соединение, при котором наблюдается стабильное соединение.

5.3 Полученные результаты

Результаты первого эксперимента представлены на рис. 3. В качестве Q-learning, DQN и UCB планировщика выступали планировщики, подробно описанные в разделе 4.

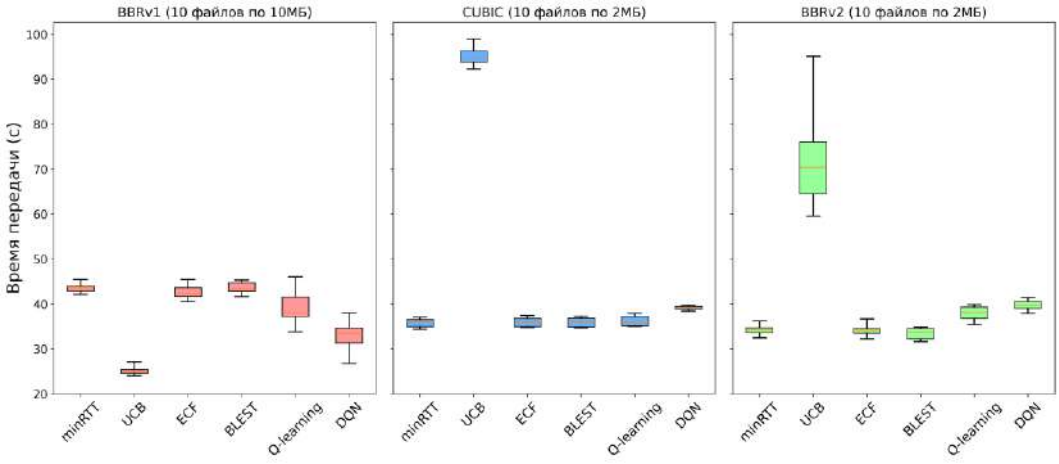


Рис. 3. Результаты экспериментов в сценарии №1.
Fig. 3. Results of experiments in scenario 1.

Планировщик UCB показал результат значительно лучше остальных в сценарии BBRv1, но в тоже время его эффективность в двух других сценариях оказалась крайне низкой. Отчасти это связано с его эвристической природой, в результате которой планировщик UCB был крайне эффективен в сценарии BBRv1 и не эффективен в двух других сценариях.

Планировщик, основанный на глубоком Q-обучении, показал результат лучше, чем minRTT, ECF и BLEST, а также Q-обучение в сценарии BBRv1. Это связано с динамически меняющимися условиями среды, так как планировщики, основанные на эвристических правилах, не всегда могут определить какой из двух путей является лучшим с точки зрения эффективности. В тоже время в сценарии CUBIC и BBRv2 планировщик, основанный на глубоком Q-обучении, показал результат немного хуже, чем minRTT, ECF и BLEST, а также Q-обучение. Это объясняется тем, что DQN был обучен в среде BBRv1.

Результаты второго эксперимента представлены на рис. 4. В качестве Q-learning, DQN и UCB планировщика выступали планировщики, подробно описанные в разделе 4.

В данном эксперименте поведение планировщиков в случае CUBIC и BBRv2 аналогично первому эксперименту. Однако результаты в случае BBRv1 несколько отличаются. Планировщик UCB работает лишь немного лучше других. Это связано с тем, что в данной среде существует путь (путь 1), который является предпочтительнее с точки зрения эффективности. Поэтому планировщики, основанные на эвристических правилах и Q-learning работают не так плохо по сравнению с UCB. В тоже время поведение и результаты планировщика, основанного на глубоком Q-обучении, аналогичны сценарию № 2.

Результаты третьего эксперимента представлены на рис. 5. В качестве Q-learning, DQN и UCB планировщика выступали планировщики, подробно описанные в разделе 4.

В данном эксперименте наблюдается примерно одинаковое поведение различных планировщиков с точки зрения эффективности. Это вполне объяснимо тем, что

характеристики путей являются идентичными и при этом каждый путь не имеет дисперсию задержек и потерь. Стоит также заметить, что планировщик, основанный на UCB, работает несколько хуже в силу эвристических правил.

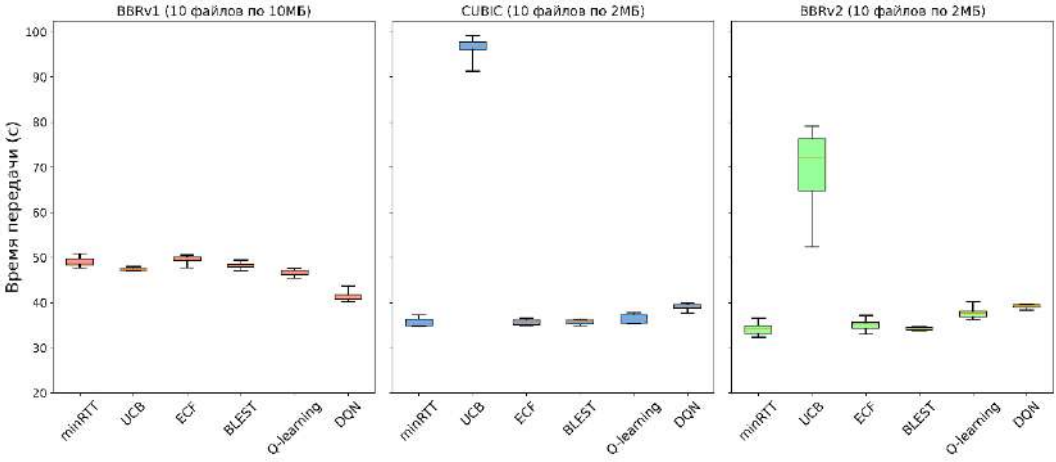


Рис. 4. Результаты экспериментов в сценарии №2.
Fig. 4. Results of experiments in scenario № 2.

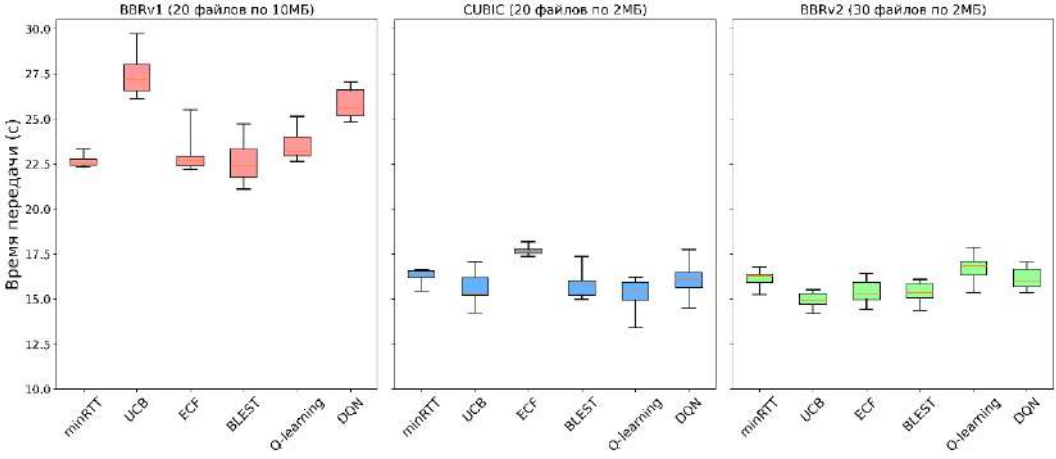


Рис. 5. Результаты экспериментов в сценарии №3.
Fig. 5. Results of experiments in scenario № 3.

5.4 Итоги

Итоговые результаты эффективности (средняя скорость в Мбит/с) планировщиков в зависимости от алгоритма контроля перегрузки представлены на рис. 6.

На основе полученных результатов необходимо отметить следующие:

1. Ни один из представленных планировщиков не оказался эффективным во всех сценариях;
2. DQN продемонстрировал самую стабильную работу;
3. UCB оказался эффективным в первом сценарии и крайне неэффективным в других;
4. Эффективность minRTT, ECF и BLEST примерно одинакова.

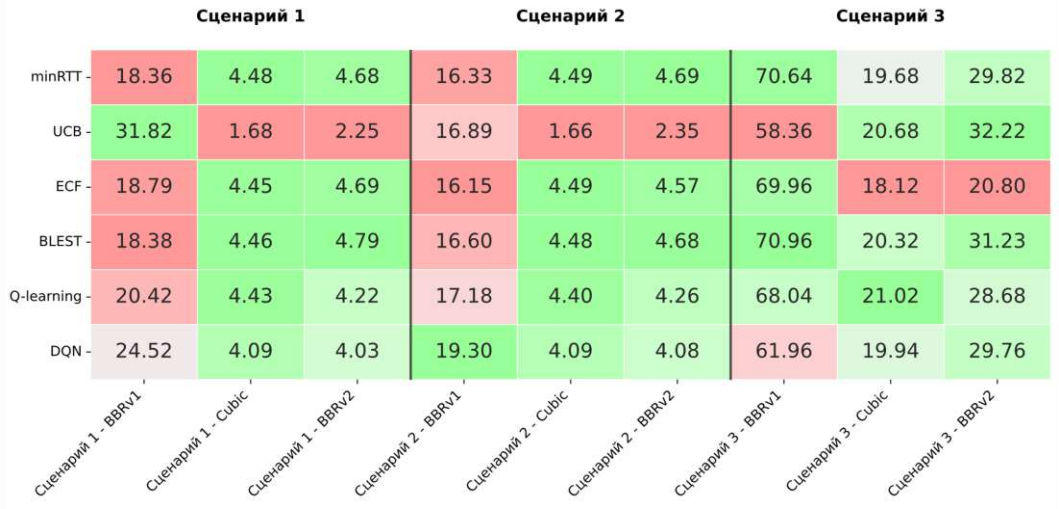


Рис. 6. Сравнительная таблица работы планировщиков.
Fig. 6. The comparative table of schedulers performance.

6. Выводы и направления будущих исследований

В ходе данной работы были исследованы и реализованы планировщики как основанные на эвристических правилах, так и на обучении с подкреплением. Реализованные планировщики исследовались не только с точки зрения характеристик путей, но и с точки зрения алгоритма контроля перегрузки. Полученные результаты говорят о том, что планировщик может эффективно работать в сетевой среде с определённым алгоритмом контроля перегрузки, но при этом быть неэффективным в среде с другим алгоритмом контроля перегрузки. Решение данной проблемы является открытым и представляет собой направление будущих исследований: разработка планировщика, работающего эффективно независимо от используемого алгоритма контроля перегрузки.

Список литературы / References

[1]. Wischik D. et al. Design, implementation and evaluation of congestion control for multipath {TCP} // 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11). 2011.
[2]. Nguyen K. et al. An approach to reinforce multipath TCP with path-aware information // Sensors. 2019. Vol. 19, issue 3, p. 476.
[3]. Chao L. et al. A brief review of multipath tcp for vehicular networks // Sensors. 2021. Vol. 21, issue 8, p. 2793.
[4]. De Coninck Q., Bonaventure O. Multipath quic: Design and evaluation // Proceedings of the 13th international conference on emerging networking experiments and technologies. 2017, pp. 160-166..
[5]. Viernickel T. et al. Multipath QUIC: A deployable multipath transport protocol // 2018 IEEE International Conference on Communications (ICC). IEEE, 2018, pp. 1-7.
[6]. Floyd S., Henderson T. The NewReno modification to TCP's fast recovery algorithm. 1999. Issue rfc2582.
[7]. Ha S., Rhee I., Xu L. CUBIC: a new TCP-friendly high-speed TCP variant // ACM SIGOPS operating systems review. 2008. Vol. 42, issue 5, pp. 64-74.
[8]. Cardwell N. et al. Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time // Queue. 2016. Vol. 14, issue 5, pp. 20-53.
[9]. Simone Ferlin и др. “BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks”. In: 2016 IFIP networking conference (IFIP networking) and workshops. IEEE. 2016, pp. 431-439.
[10]. Yeon-sup Lim и др. “ECF: An MPTCP path scheduler to manage heterogeneous paths”. In: Proceedings of the 13th international conference on emerging networking experiments and technologies. 2017, pp. 147-159.

[11]. Wu H. et al. Peekaboo: Learning-based multipath scheduling for dynamic heterogeneous environments // IEEE Journal on Selected Areas in Communications. 2020. Vol. 38, issue 10, pp. 2295-2310.
[12]. Wu H. et al. Multipath scheduling for 5G networks: Evaluation and outlook // IEEE Communications Magazine. 2021. Vol. 59, issue 4. pp. 44-50.
[13]. Lee S., Yoo J. Reinforcement learning based multipath QUIC scheduler for multimedia streaming // Sensors. 2022. Vol. 22, issue 17, p. 6333.
[14]. Roselló M. M. Multi-path scheduling with deep reinforcement learning // 2019 European Conference on Networks and Communications (EuCNC). IEEE, 2019, pp. 400-405.
[15]. Wu H. et al. Falcon: Fast and accurate multipath scheduling using offline and online learning // arXiv preprint arXiv:2201.08969. 2022.
[16]. Mininet [Электронный ресурс]. – URL: https://mininet.org/ (дата обращения: 15.05.2025).
[17]. XQUIC [Электронный ресурс]. – URL: https://github.com/alibaba/xquic (дата обращения: 15.05.2025).

Информация об авторах / Information about authors

Максим Владимирович ПОПОВ – старший лаборант ИСП РАН, студент магистратуры факультета ВМК МГУ. Сфера научных интересов: анализ сетевого трафика, алгоритмы контроля перегрузок.

Maxim Vladimirovich POPOV – graduate student at the Faculty of Computational Mathematics and Cybernetics of Moscow State University. Research interests: network traffic analysis, congestion control algorithms.

Иван Александрович СТЕПАНОВ – аспирант ИСП РАН, стажёр-исследователь ИСП РАН, ассистент кафедры информатики и вычислительной математики МФТИ. Сфера научных интересов: анализ сетевого трафика с помощью машинного обучения.

Ivan Alexandrovich STEPANOV – postgraduate student of the ISP RAS, intern researcher at ISP RAS, an assistant at the Department of Computer Science and Computational Mathematics at MIPT. Research interests: network traffic analysis using machine learning.

Александр Игоревич ГЕТЬМАН – кандидат физико-математических наук, старший научный сотрудник ИСП РАН, ассистент ВМК МГУ и МФТИ, доцент ВШЭ. Сфера научных интересов: анализ бинарного кода, восстановление форматов данных, анализ и классификация сетевого трафика.

Aleksandr Igorevich GETMAN – Cand. Sci. (Phys.-Math.), senior researcher at ISP RAS, assistant at CMC MSU and MIPT, associate professor at HSE. Research interests: binary code analysis, data format recovery, network traffic analysis and classification.

DOI: 10.15514/ISPRAS-2025-37(6)-17

Тестирование подсистемы безопасности ОС Astra Linux на основе формализованного описания модели управления доступом

П.Н. Девянин, ORCID: 0000-0003-2561-794X <pdevyanin@astralinux.ru>
С.С. Жиляков, ORCID: 0009-0006-1831-697X <szhiliakov@astralinux.ru>
А.И. Смирнов, ORCID: 0009-0000-3483-110X <alesmirnov@astralinux.ru>

ООО «РусБИТех-Астра»,
117105, г. Москва, Варшавское шоссе, д. 26.

Аннотация. В операционной системе (ОС) Astra Linux кроме традиционного для большинства ОС дискреционного управления доступом ее подсистемой безопасности PARSEC реализуются механизмы мандатного контроля целостности (МКЦ) и мандатного управления доступом (МРД). С учетом многообразия имеющихся в данной ОС сущностей (объектов доступа, файлов, каталогов, сокетов и др.) и субъектов (процессов) эти механизмы имеют сложную логику функционирования, затрудняющую их тестирование с использованием вручную подготовленных тестов. Влияет на проблему необходимость выполнения процессов разработки безопасного программного обеспечения (ПО) для соответствия ОС Astra Linux требованиям высших классов защиты и уровней доверия. Вместе с тем в основе механизмов МКЦ и МРД этой ОС используется мандатная сущностно-ролевая ДП-модель управления доступом и информационными потоками в ОС семейства Linux (МРОСЛ ДП-модель), описанная в классической математической нотации и в формализованной нотации на языке формального метода Event-B. Авторами развивается рекомендованный ГОСТ Р 59453.4-2025 подход к тестированию механизмов управления доступом на основе сбора трасс системных вызовов ОС и их перевода на язык формальной модели с целью проверки соответствия ей логики функционирования механизма управления доступом ОС. Результатам этой работы посвящена настоящая статья, в которой, во-первых, изложены итоги разработки и верификации используемого для тестирования нижнеуровневого представления МРОСЛ ДП-модели (PARSEC-модели), выполненного на языке формального метода Event-B и представляющего функциональную спецификацию связанных с управлением доступом системных вызовов ОС. Во-вторых, описывается система тестирования, включающая модуль ядра ОС для сбора трасс системных вызовов, ПО для их преобразования в модельные трассы, аниматор модельных трасс, выполненный с применением инструментального средства ProB, и ПО для формирования результатов тестирования в формате инструментального средства Allure. В-третьих, в статье рассматривается подход к использованию для распараллеливания тестирования технологии eBPF.

Ключевые слова: операционная система; операционная система Astra Linux; МРОСЛ ДП-модель; метод Event-B; тестирование.

Для цитирования: Девянин П.Н., Жиляков С.С., Смирнов А.И. Тестирование подсистемы безопасности ОС Astra Linux на основе формализованного описания модели управления доступом. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025, стр. 21–36. DOI: 10.15514/ISPRAS–2025–37(6)–17.

Testing the Astra Linux OS Security Subsystem Based on a Formalized Description of the Access Control Model

P.N. Devyanin, ORCID: 0000-0003-2561-794X <pdevyanin@astralinux.ru>
S.S. Zhiliakov, ORCID: 0009-0006-1831-697X <szhiliakov@astralinux.ru>
A.I. Smirnov, ORCID: 0009-0000-3483-110X <alesmirnov@astralinux.ru>

RusBITech-Astra,
26, Varshavskoe, Moscow, 117105, Russia.

Abstract. In Astra Linux operating system (OS), in addition to the traditional Discretionary Access Control, its PARSEC security subsystem implements Mandatory Access Control policies such as Mandatory Integrity Control (MIC) and Multilevel Security (MLS). Given the variety of entities (objects, files, directories, sockets, etc.) and subjects (processes) available in a given OS, these policies often have a complicated logic of functioning, which makes it difficult to test them using manual testing. This problem is especially aggravated in the context of the necessity to fulfill the Security Development Lifecycle in compliance with the requirements of the highest protection classes and trust levels established by the FSTEC of Russia regulatory documents. Besides, the MIC and MLS policies of this OS are based on the mandatory entity-role model of access and information flows security control in OS of Linux family (MROSL DP-model), described in the classical mathematical notation and in the formalized notation using the formal Event-B method. Therefore, the authors of this paper have developed and finalized the approach recommended by GOST R 59453.4-2025, taking into account the specifics of current releases of Astra Linux OS, which consists of tracing system calls and translating them into the language of the formal model in order to verify the compliance of the functioning access control policies with the model. The results of this work are described in this paper, which, firstly, outlines the results of the development and verification of the so-called lower-level representation of the MROSL DP-model (PARSEC-model) used for testing, performed in the Event-B and in essence represents a functional specification of access control-related system calls of the OS. Secondly, it describes a testing system that includes the Linux Kernel Module for tracing system calls, software for their translation into model traces, an animator of model traces using the ProB toolkit, and software for generating test results in the format of the Allure toolkit. Thirdly, the paper considers approach to using eBPF technology for parallelizing testing.

Keywords: operating system; Astra Linux; MROSL DP-model; Event-B; testing.

For citation: Devyanin P.N., Zhiliakov S.S., Smirnov A.I. Testing the Astra Linux OS security subsystem based on a formalized description of the access control model. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025. pp. 21-36 (in Russian). DOI: 10.15514/ISPRAS-2025-37(6)-17.

1. Введение

Тестирование программного обеспечения (ПО) на выявление отличий между его реально существующими и требуемыми свойствами (функциональное тестирование) согласно ГОСТ Р 56939-2024 [1] является неотъемлемой частью процесса разработки безопасного ПО. Реализация этого процесса особенно важна, когда ПО выполняет функции средства защиты информации (СЗИ), например, являясь операционной системой (ОС). Ключевую роль в таких СЗИ часто выполняет механизм управления доступом. Для большинства ОС это дискреционное управление доступом [2-3]. Такой механизм достаточно прост для реализации и понимания, его тестирование, разработка соответствующих тестов, как правило, не представляют затруднений. Кроме того, в ОС семейств Linux или Windows он не претерпевал существенных изменений в течение десятилетий.

Вместе с тем, задача тестирования механизма управления доступом ОС существенно усложняется, когда в него включают реализацию других политик управления доступом таких, как мандатный контроль целостности (МКЦ) или мандатное управление доступом (МРД). Примером ОС с МКЦ и МРД является сертифицированная по установленным нормативными документами ФСТЭК России [4] требованиям высших классов защиты и уровней доверия ОС Astra Linux (операционная система специального назначения Astra Linux

Special Edition) [5-6]. Как во многих ОС семейства Linux в этой ОС управление доступом осуществляется между субъектами (процессами), функционирующими от имени десятков или даже сотен локальных, системных или зарегистрированных в домене учетных записей пользователей, к широкому спектру сущностей (объектов доступа, файлов, каталогов, сокетов и др.). При этом субъектам, учетным записям пользователей назначаются уровни целостности и уровни доступа, используемые МКЦ и МРД привилегии, а сущностям – уровни целостности и уровни доступа, специальные флаги. Все перечисленное затрудняет понимание логики функционирования МКЦ и МРД, их тестирование с использованием вручную подготовленных тестов.

Научной основой реализации МКЦ и МРД в ОС Astra Linux является соответствующая критериям ГОСТ Р 59453.1-2021 [2] мандатная сущностно-ролевая ДП-модель управления доступом и информационными потоками в ОС семейства Linux (МРОСЛ ДП-модель) [3]. Данная модель описана в двух нотациях: математической (аналогично классическим моделям [7]) и, что наиболее важно в контексте настоящей статьи, в формализованной на языке формального метода Event-B [8-10]. Машиночитаемый язык представления модели в этой нотации обеспечивает возможность выполнения рекомендаций ГОСТ Р 59453.4-2025 [11] и применения апробированного подхода [12-13] по автоматизированному тестированию механизма управления доступом ОС, заключающегося в сборе трасс системных вызовов ОС, их переводе на язык формальной модели с целью проверки соответствия ей логики функционирования этого механизма. Такой подход, во-первых, отвечает реализуемой «Группой Астра» (ООО «РусБИТех-Астра») единой методологии разработки безопасного системного ПО [14], во-вторых, сокращает ресурсы на тестирование МКЦ и МРД по мере расширения их функций, изменения их программного кода, а, в-третьих, позволяет учитывать модификации, вносимые в МРОСЛ ДП-модель, которая согласно рекомендациям ГОСТ Р 59453.3-2025 [15-16] также регулярно дорабатывается и развивается [17].

Следует отметить, что непосредственно соответствующее описанию МРОСЛ ДП-модели в математической нотации ее описание в формализованной нотации (названное верхнеуровневым) позволяет согласно рекомендациям ГОСТ Р 59453.2-2021 [18] дедуктивно верифицировать модель с применением инструментального средства Rodin [10, 19]. Однако использование такого верхнеуровневого описания модели напрямую для задачи автоматизированного тестирования не является возможным в силу его абстрактности, а, именно, в нем не обеспечивается точное соответствие программному коду системных вызовов ядра ОС и подсистемы безопасности PARSEC, реализующей МКЦ и МРД в ОС Astra Linux. В связи с этим на языке формального метода Event-B было разработано нижнеуровневое представление модели (названное PARSEC-моделью), которое позволило применить подход, изложенный в [12-13], и использовать разработанные его авторами инструментальные средства. В том числе, для монитора системных вызовов был применен специальный модуль ядра ОС, использующий механизмы инструментации исполняемого кода Kprobes, Kretprobes и отправляющий полученные им данные мониторинга в пользовательское пространство ОС для дальнейшего их преобразования в модельные трассы и их анимации с применением инструментального средства ProB [20].

Вместе с тем, данный подход имеет некоторые недостатки. Во-первых, разработка модулей ядра ОС всегда сопряжена с риском – ошибка в его программном коде может значительно повлиять на поведение ОС, в том числе привести к ее краху. Во-вторых, такие модули требуют компиляции под каждую версию ядра ОС, что значительно усложняет процесс автоматизированного тестирования ОС с несколькими поддерживаемыми версиями ядра (к таким ОС относится и ОС Astra Linux). В-третьих, монитор системных вызовов не обладает функцией одновременного сбора их трасс для нескольких процессов, что замедляет тестирование и затрудняет развертывание тестовой инфраструктуры в среде непрерывной разработки и интеграции ОС (CI). Наконец, текущая реализация монитора системных вызовов не отслеживает обращения прикладного ПО к подсистеме безопасности PARSEC

при изменении уровней конфиденциальности и целостности (меток безопасности) компонент ОС.

В связи с изложенным авторами статьи был рассмотрен подход к применению технологии eBPF, представляющей собой альтернативу загружаемым модулям ядра. Данная технология ранее применялась для трассировки сетевых пакетов [21], и для анализа производительности ОС [22]. В настоящее время eBPF также применяется в таких системах мониторинга средств виртуализации на уровне ядра ОС, как Tetragon, Falco и Tracee [23-25].

Таким образом, статья организована следующим образом. В следующем разделе рассматриваются результаты разработки и верификации нижнеуровневого представления МРОСЛ ДП-модели (PARSEC-модели). В разд. 3 описывается система тестирования подсистемы безопасности PARSEC ОС Astra Linux. В разд. 4 анализируется подход к применению технологии eBPF для расширения функциональных возможностей этой системы тестирования, в том числе за счет его распараллеливания. Заключение завершает статью, в нем подводятся итоги выполненных исследований и разработок, а также рассматриваются дальнейшие направления их развития.

2. Нижнеуровневое представление МРОСЛ ДП-модели (PARSEC-модель)

МРОСЛ ДП-модель является основой для реализации механизма управления доступом в ОС Astra Linux. В математической и формализованной нотациях модель имеет иерархическое представление и включает согласованное описание ролевого управления доступом (РУД, представляющего штатное для ОС семейства Linux дискреционное управление доступом), МКЦ и МРД [3]. В обеих нотациях согласно ГОСТ Р 59453.1-2021 [2] модель включает описание состояний и правил перехода между состояниями абстрактного автомата, соответствующего политикам управления доступом, реализуемым в ОС Astra Linux.

Как уже было отмечено, непосредственно соответствующее описанию МРОСЛ ДП-модели в математической нотации ее верхнеуровневое представление в формализованной нотации на языке формального метода Event-B достаточно абстрактно. В нем не учитываются некоторые особенности программной реализации управления доступом в ядре ОС и подсистеме безопасности PARSEC, являющейся модулем безопасности ядра – Linux Security Module (LSM) [26]. Например, они оперируют не абстрактными сущностями (объектами доступа) и субъектами, а файловыми дескрипторами, индексными дескрипторами (inode) и другими системными компонентами ОС. Также реализованные в ядре ОС обработчики системных вызовов (выполняющих операции над файлами, каталогами, процессами, учетными записями пользователей) по своему функционалу отличаются от их описания в соответствующих правилах преобразования состояний абстрактного автомата в верхнеуровневом представлении модели.

Чтобы приблизить описание модели в формализованной нотации к программной реализации управления доступом в ядре ОС и подсистеме безопасности PARSEC, а также создать условия для применения этого описания в системе автоматизированного тестирования, на Event-B было разработано нижнеуровневое представление МРОСЛ ДП-модели (PARSEC-модель), которое по сути является функциональной спецификацией обработчиков системных вызовов в ядре ОС и подсистеме безопасности PARSEC. Выбор Event-B позволил, во-первых, сохранить преемственность технологий описания и верификации модели с использованными для ее верхнеуровневого представления, а, во-вторых, применить подход, изложенный в [12-13], который также базируется на Event-B.

PARSEC-модель аналогично верхнеуровневому представлению с применением техники пошагового уточнения (refinement) Event-B [8] имеет иерархическое представление, включающее три уровня, соответствующие ключевым для режимов защищенности ОС Astra Linux политикам управления доступом [6]:

- Первый уровень – «Базовый» («Орел»), дискреционное управление доступом;
- Второй уровень – «Усиленный» («Воронеж»), мандатный контроль целостности;
- Третий уровень – «Максимальный» («Смоленск»), мандатное управление доступом.

Каждый уровень представлен соответствующими контекстами (contexts), описывающими статические компоненты моделируемой ОС, и машинами (machines), включающими описания соответствующих системным вызовам событий (events) и инвариантов безопасности (invariants).

PARSEC-модель содержит описание следующих событий (системных вызовов): open, openat, mkdir, chmod, fchmod, fchmodat, chown, fchown, fchownat, chdir, fchdir, getxattr, setxattr, link, execve, umask, exit, close, unlink, rmdir, fork. В ней также присутствует описание события chlbl, которое соответствует функции sys_chlbl, входящей в интерфейс подсистемы безопасности PARSEC и позволяющей изменять уровни целостности и конфиденциальности сущности (файла или каталога).

Стоит отметить, что на текущий момент РУД, описанное в верхнеуровневом представлении МРОСЛ ДП-модели, явно не реализовано в ОС, и поэтому его формализация отсутствует в нижнеуровневой PARSEC-модели. Кроме того, в нем отсутствуют некоторые несущественные, но сложные для моделирования элементы дискреционного управления доступом (например, расширенные списки контроля доступа Access Control Lists – ACL), поскольку они не интересны при тестировании подсистемы безопасности PARSEC.

Т.к. наибольшее значение при тестировании управления доступом в ОС Astra Linux представляют ее собственные механизмы защиты, то далее будут рассмотрены результаты разработки второго и третьего уровней PARSEC-модели, соответствующих МКЦ и МРД, тем более, что большая часть первого уровня была описана в рамках исследований [12-13].

Первым шагом при моделировании МКЦ в PARSEC-модели стало описание уровней целостности. В ОС программная реализация каждого уровня целостности состоит из двух частей: неиерархического уровня (категорий) целостности (маски 32 бит) и линейного уровня целостности (знаковых целых чисел от -128 до 127). Например, уровень целостности 0x00000002:-128 соответствует неиерархическому уровню «Виртуализация» 0x00000002 (в соответствующей маске бит ненулевой только второй бит) и минимальному линейному уровню -128. Поэтому в модели для субъектов (процессов), сущностей (файлов или каталогов) и учетных записей пользователей заданы по паре соответствующих функций, например, функции EntityInt и EntityLinearInt для сущностей (Листинг 1).

Поскольку формальный метод Event-B не поддерживает битовые операции над целыми числами, то в PARSEC-модели множество неиерархических уровней целостности Integrity задано как множество всех подмножеств несущего множества (carrier set) Int, элементы которого представляют собой позиции битов в двоичном представлении целого числа. Такое описание позволяет выразить программную реализацию битовой маски в терминах теории множеств, где каждому биту будет соответствовать нумерованная константа из множества Int. Например, значению неиерархической уровня целостности 0x0000003f (двоичному значению 0b111111, а десятичному значению 63 – максимальному по умолчанию уровню целостности max_ilev в ОС Astra Linux) будет соответствовать множество констант {Int1, Int2, Int3, Int4, Int5, Int6} ∈ Integrity. Сравнение любых двух неиерархических уровней целостности в таком случае естественно выражается с помощью операций из теории множеств. Так как в ОС значение линейного уровня целостности ограничено одним байтом и является знаковым числом, метод Event-B позволяет определить тип такой константы как интервал -128 .. 127, и сравнение линейных уровней целостности в данном случае будет соответствовать сравнению целых чисел.

Вторым шагом при моделировании МКЦ в PARSEC-модели стало описание инвариантов состояний. Для примера рассмотрим инвариант, который устанавливает требования к уровням

целостности субъекта и сущности при доступе на запись (Листинг 2). В этом фрагменте используются следующие функции:

- SubjectFDs – задает для процесса его файловые дескрипторы;
- FDFile – отображает файловый дескриптор на соответствующий файл;
- FDFlags – задает для файлового дескриптора флаги, с которыми был открыт соответствующий файл.

```
sets
  Int
  ENTITIES
constants
  Integrity
  LinearIntegrity
axioms
  @Integrity_type:  Integrity = ℙ(Int)
  @LinearIntegrity_type:  LinearIntegrity = -128..127
variables
  Entities
  EntityInt
  EntityLinearInt
invariants
  @Entities_type:              Entities ⊆ ENTITIES
  @EntityInt_type:             EntityInt ∈ Entities → Integrity
  @EntityLinearInt_type: EntityLinearInt ∈ Entities → LinearIntegrity
```

Листинг 1. Функции EntityInt и EntityLinearInt.
Listing 1. EntityInt и EntityLinearInt functions.

```
@SubjectAccessObject_Write:
  ∀ s,fd   s ↦ fd ∈ SubjectFDs ∧ FDFile(fd) ∈ Entities \ Containers ∧
  (O_WRONLY ∈ FDFlags(fd) ∨ O_RDWR ∈ FDFlags(fd)) ⇒
  EntityInt(FDFile(fd)) ⊆ SubjectInt(s) ∧ EntityLinearInt(FDFile(fd))
                        ≤ SubjectLinearInt(s)
```

Листинг 2. Инвариант установки требований к уровням целостности.
Listing 2. Invariant for setting integrity level requirements.

В ОС факт доступа на запись процесса к некоторому файлу можно определить по наличию дескриптора этого файла в множестве файловых дескрипторов этого процесса, а также наличия флагов O_WRONLY или O_RDWR, с которыми был открыт этот файл. Кроме того, т.к. ядро ОС ограничивает получение файлового дескриптора к каталогу с доступом на запись (в таком случае выдается ошибка EISDIR), то рассматриваемый инвариант включает только проверку доступов на запись процесса к файлам. Таким образом, инвариант позволяет выразить проверяемое в ОС условие согласованности уровней целостности процесса и файла при наличии к нему доступа на запись – уровень целостности процесса должен быть не ниже уровня целостности файла.

После контекста и инвариантов были описаны события (системные вызовы). Для удобства верификации PARSEC-модели и ее использования в системе тестирования для некоторых системных вызовов ОС задавались по два события. Например, для системного вызова sys_chlbl были описаны события: chlbl_container – для изменения уровня целостности сущности-контейнера (каталога), chlbl_object – для изменения уровня целостности сущности-объекта (файла). Такое разделение обусловлено наличием у каталогов и файлов специфичных параметров МКЦ. В их числе флаг SILEV исполняемого файла, который задает порядок активизации из него процесса с учетом уровней целостности этого файла и учетной записи

пользователя, от имени которой активизируется процесс (это позволяет процессам в сессии учетной записи пользователя с низким уровнем целостности запускать некоторые процессы, которым для выполнения их функций необходим высокий текущий уровень целостности; так с помощью флага SILEV помечается исполняемый файл консольной утилиты passwd, запускаемой для изменения пароля низкоцелостной учетной записи пользователя, образ которого хранится в обладающем высоким уровнем целостности файле «/etc/shadow»). Поэтому событие chlbl_container содержит специальную переменную passed_silev, используемую для предотвращения установки флага SILEV на каталоги, и соответствующие два охранные условия (guard):

```
@grd10: passed_silev ∈ BOOL
@grd11: passed_silev = FALSE
```

Таким образом, если в системе тестирования на этапе трансляции системной трассы в события PARSEC-модели значение переменной passed_silev будет установлено TRUE (что в ОС соответствует тому, что на каталог делается попытка установить флаг SILEV), то во время анимации трассы условие @grd11 не будет выполнено, что будет сигнализировать о расхождении программной реализации подсистемы безопасности PARSEC и PARSEC-модели. Аналогично при моделировании МКЦ описываются другие флаги файлов (SSI) и каталогов (IINH, IRELAX, SSI) [27].

Для выполнения инвариантов безопасности в PARSEC-модели в событиях описаны соответствующие охранные условия. Так истинность рассмотренного инварианта @SubjectAccessObject_Write в событии открытия сущности (файла или каталога) open_exists обеспечивается охранным условием, показанным в Листинге 3.

```
@grd24:
  (O_WRONLY ∈ flags ∨ O_RDWR ∈ flags) ⇒
  (file ∈ Containers ∧ IRELAX(file) = TRUE) ∨
  ((file ∈ Entities \ Containers ∨ (file ∈ Containers ∧ IRELAX(file) = FALSE)) ∧
   EntityInt(file) ⊆ SubjectInt(proc) ∧
   EntityLinearInt(file) ≤ SubjectLinearInt(proc)
  )
```

Листинг 3. Охранное условие открытия файла.
Listing 3. File open guard condition.

Третий уровень PARSEC-модели соответствует МРД. В его рамках уровни конфиденциальности (доступа) сущностей, субъектов и учетных записей пользователей были заданы аналогично уровням целостности, т. к. они также состоят из двух частей: линейных уровней конфиденциальности и неиерархических уровней (категорий) конфиденциальности. Также аналогично уровню МКЦ заданы специальные флаги сущностей (CCNR, EHOLE, WHOLE) [6].

При описании инвариантов состояний акцентировалось внимание на те из них, которые отражают МРД. Например, в любой момент функционирования ОС никакой процесс не должен иметь доступ на чтение к файлу или каталогу с уровнем конфиденциальности большим, чем уровень доступа процесса (за исключением случаев наличия у процесса либо специальных привилегий PARSEC_CAP_IGNMACLVL и PARSEC_CAP_IGNMACCAT, объединенных в PARSEC-модели в множество PARSEC_CAP_IGNMAC, либо полномочий суперпользователя root и максимального уровня целостности HighI, позволяющих ему нарушать МРД). Иллюстрация дана на Листинге 4.

Также для примера выполнение этого инварианта в событии открытия сущности (файла или каталога) open_exists было обеспечено охранным условием, показанным на Листинге 5.

В итоге разработки PARSEC-модели, выполнение всех инвариантов состояний было дедуктивно доказано при верификации с применением инструментального средства Rodin [19].

```
@SubjectAccessesConf_Read:
  ∀s,fd · s ↦ fd ∈ SubjectFDs ∧ FDFile(fd) ∈ Entities ∧ s ∈ Subjects ∧
   (O_RDONLY ∈    FDFlags(fd) ∨ O_RDWR ∈ FDFlags(fd)) ∧
  ¬(PARSEC_CAP_IGNMAC ⊆ SubjectCaps(s)) ∧
  ¬(SubjectUser(s) = ROOT_USER ∧ SubjectInt(s) = HighI ∧ SubjectLinearInt(s) ≥ 0)
  ⇒ (EntityLev(FDFile(fd)) ≤ SubjectLev(s) ∧ EntityCat(FDFile(fd)) ⊆ SubjectCat(s))
```

Листинг 4. Проверка прав доступа и наличия привилегий.
Listing 4. Checking access rights and privileges.

```
@grd26: (O_RDONLY ∈ flags ∨ O_RDWR ∈ flags) ⇒
      (EntityLev(file) ≤ SubjectLev(proc) ∧ EntityCat(file) ⊆ SubjectCat(proc)) ∨
            (PARSEC_CAP_IGNMAC ⊆ SubjectCaps(proc)) ∨
      (SubjectUser(proc) = ROOT_USER ∧ SubjectInt(proc)=HighI ∧
       SubjectLinearInt(proc) =0)
```

Листинг 5. Охранное условие открытия сущности (файла или каталога).
Listing 5. Entity (file or directory) open guard condition.

В заключении раздела отметим, что в [28] было сказано о потенциальной возможности разработки PARSEC-модели путем уточнения напрямую верхнеуровневого представления МРОСЛ ДП-модели в формализованной нотации. Однако такой подход кратно увеличивает сложность модели. Поэтому при разработке PARSEC-модели авторами настоящей статьи был использован экспертный подход, включающий в себя построение модели на основе анализа верхнеуровневого представления МРОСЛ ДП-модели в обеих нотациях, а также анализа исходного кода ядра ОС Astra Linux и подсистемы безопасности PARSEC.

3. Система тестирования подсистемы безопасности PARSEC ОС Astra Linux

Система тестирования (рис. 1) реализует автоматическую проверку соответствия поведения подсистемы безопасности PARSEC и нижнеуровневого представления МРОСЛ ДП-модели – PARSEC-модели. При этом за ее основу была взята система, описанная в [12-13].

Применение системы тестирования включает выполнение следующих шагов:

1. Подготовка начального окружения системы тестирования, включающая настройку ОС Astra Linux, создание соответствующих файлов, каталогов, учетных записей пользователей, задание их параметров МКЦ и МРД.
2. Сбор и сохранение инструментом gather_info информации о начальном окружении системы, которое было сформировано на шаге 1.
3. Загрузка модуля monitor в пространство ядра, и запуск вспомогательного процесса umonitor для взаимодействия с ним.
4. Запуск теста, включающего воздействия на подсистему безопасности PARSEC, и запись результатов мониторинга тестируемого процесса в базу данных (получение системной трассы тестируемого процесса).
5. Создание модельной трассы с помощью инструмента mediator из системной трассы, полученной на шаге 4, и начального окружения, полученного на шаге 2.
6. Анимация трассы, полученной на шаге 5, с использованием PARSEC-модели и инструментального средства ProB [20], преобразование ее результатов в формат платформы Allure [29] с помощью скрипта allure_report.py.
7. Получение вывода о соответствии поведения тестируемого процесса в ОС Astra Linux и результатами анимации на шаге 6 его модельной трассы.

В контексте настоящего исследования по сравнению с [12-13] были доработаны ряд шагов. Так при сборе данных о начальном окружении системы на шаге 2 важной составляющей являются значения специальных флагов МКЦ и МРД, уровней целостности и

конфиденциальности (доступа) файлов и каталогов, учетных записей пользователей (на этом этапе не собираются параметры процессов). Для этого данные о значениях флагов, уровней целостности и конфиденциальности файла или каталога извлекаются из их расширенных атрибутов; а для учетной записи пользователя – из соответствующих ее идентификатору uid файлов в каталогах /etc/parsec/micdb (уровень целостности) и /etc/parsec/macdb (уровень доступа).

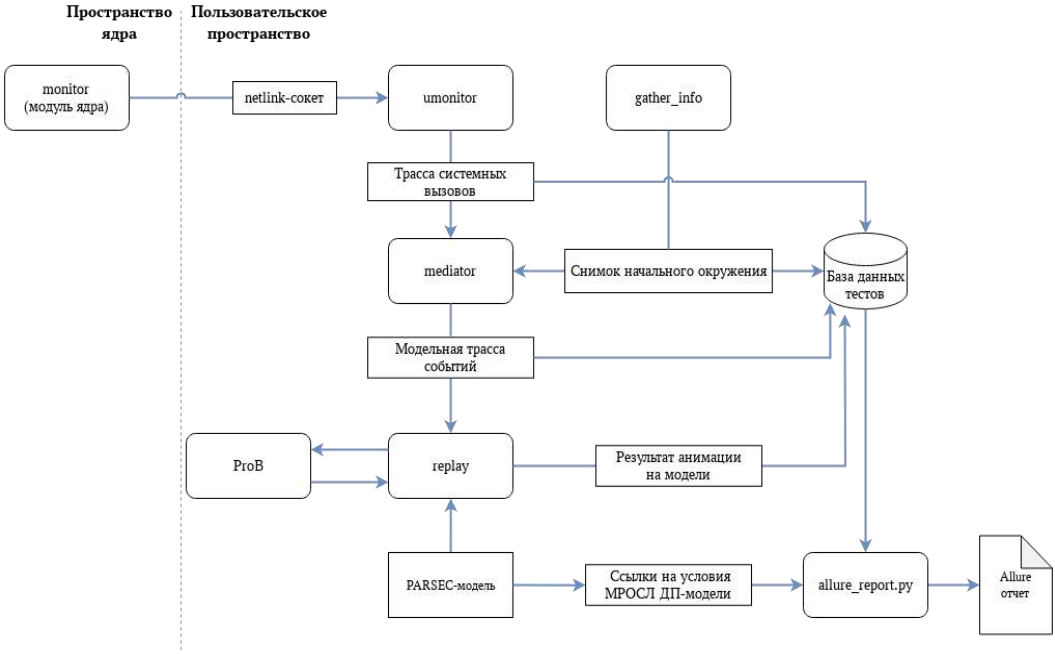


Рис. 1. Схема системы тестирования.
Fig. 1. Schematic diagram of the testing system.

Мониторинг на шаге 4 осуществляется с помощью модуля ядра ОС с применением механизмов инструментации исполняемого кода Kprobes и Kretprobes. Примечательным для рассматриваемой системы является то, что помимо стандартных системных вызовов был реализован перехват функции подсистемы безопасности PARSEC – sys_chlbl. Как было отмечено в предыдущем разделе, эта функция является интерфейсом для изменения уровней целостности и конфиденциальности файлов и каталогов. Несмотря на функциональное сходство системных вызовов sys_setxattr (предназначен для установки расширенных атрибутов) и sys_chlbl, назначение последнему отдельного события chlbl упрощает верификацию и делает структуры PARSEC-модели более понятными.

Инструмент трансляции трасс на шаге 5 преобразует системные трассы в модельные с учётом особенностей ОС Astra Linux и PARSEC-модели:

- битовые маски, соответствующие неиерархическим уровням (категориям) целостности и конфиденциальности, отображаются в их заданные с помощью множеств эквиваленты;
- специальные флаги интерпретируются как имеющие булевы значения;
- события выбираются в зависимости от типа обрабатываемой сущности – файл или каталог.

На этапе трансляции особое внимание уделяется обработке компонент из начального окружения системы. Создание этих компонент на шаге 1 не отображается в виде

последовательности системных вызовов в системной трассе, и информацию о том как они были созданы и сконфигурированы необходимо выводить из их результирующего по итогу выполнения шага состояния. Например, пусть при создании файла в каталоге в начальном окружении присутствует каталог /foo и файл /foo/bar с максимальными уровнями целостности HighI (в ОС Astra Linux ему соответствует системное значение max_ilev, по умолчанию равное 0x0000003f:0, т.е. десятичному значению 63). Также пусть на каталог /foo/bar не установлены никакие специальные флаги. При ошибочной трансляции можно предположить, что файл /foo/bar был создан с уровнем целостности HighI, наследуя его от родительского каталога. Однако такое наследование возможно только в случае, если на каталог был установлен специальный флаг IINH. Таким образом, событие создания файла /foo/bar необходимо рассматривать как последовательность двух событий open_create (системный вызов open, создающий файл) и chlbl_object (устанавливает на файл уровень целостности HighI).

Как отмечено в работах [10, 30] ProB как инструментальное средство, используемое для анимации трасс на шаге 6, имеет ряд ограничений, одно из которых – невозможность работы с множествами большой мощности (из-за проблемы «комбинаторного взрыва»). Такое ограничение не позволяет использовать в PARSEC-модели несущие множества, точно соответствующие неиерархическим уровням (категориям) целостности и конфиденциальности, поскольку в программной реализации ОС они представлены в виде масок размером 32 и 64 бита, соответственно. Поэтому на шаге 5 для дальнейшей корректной анимации модельных трасс используются урезанные несущие множества для неиерархических уровней (категорий) целостности и конфиденциальности – 8 бит.

Сами тесты для шага 4 были сформированы в соответствии с верхнеуровневой МРОСЛ ДП-моделью в математической нотации, т. к. именно в ней формализовано описание МКЦ и МРД, реализуемое в ОС Astra Linux. В основе системы тестирования используется фреймворк pytest [31], который позволяет автоматизировать ее конфигурирование, запуск, а также «очистку» после очередного теста. Каждый тест представляет собой исполняемый скрипт на языке Python и задает описание последовательности действий в ОС (создание, удаление, чтение, запись, выполнение, изменение уровней целостности и конфиденциальности файлов и каталогов). В настоящее время полнота тестирования в основном обеспечивается экспертным анализом МРОСЛ ДП-модели, но в будущем планируется создавать тесты с применением формального описания PARSEC-модели и оценки покрытия ее инвариантов и охранных условий событий. Такой подход соответствует рекомендациям по оценке покрытия формальных моделей ГОСТ Р 59453.4-2025 [11] и включает в себя создание тестов с использованием структурных элементов спецификации событий модели. Например, рассмотренное в предыдущем разделе охранное условие @grd26 события open_exists необходимо разбить на дизъюнкты и обеспечить минимальное тестовое покрытие, при котором каждый дизъюнкт хотя бы раз принимает значение истина (TRUE).

На шаге 6 данные о проведенном тестировании преобразуются в формат платформы Allure [29], для чего результат каждого теста транслируется в специальный JSON-объект.

При анализе результатов тестирования на шаге 7 наибольший интерес представляют тесты, завершившиеся неудачей. Для его осуществления отчет о каждом таком тесте должен в явном виде указывать на событие и охранные условия (guards), которые в нем были нарушены. Для упрощения работы с PARSEC-моделью большинство охранных условий имеют ссылку на соответствующие условия в МРОСЛ ДП-модели в математической нотации. Таким образом, для тестов, завершившихся неудачей, в отчете Allure указываются следующие данные: системная и модельная трассы; имя события в PARSEC-модели; идентификатор нарушенного охранного условия; предикат, соответствующий охранному условию; комментарий, указывающий на то, какое условие МРОСЛ ДП-модели было нарушено. Это позволяет упростить анализ ошибок, т.к. он не требует глубокого «погружения» в детали описания всей PARSEC-модели.

4. Применение технологии eBPF для распараллеливания тестирования

В основе системы тестирования, рассмотренной в предыдущем разделе, лежит модуль monitor ядра ОС, осуществляющий мониторинг тестируемых процессов с применением механизмов Kprobes и Kretprobes. Он перехватывает выполнение функций ядра ОС, что позволяет собирать данные об этих функциях, включая аргументы, с которыми они были вызваны. Для взаимодействия с пользовательским пространством ОС этот модуль ядра использует Netlink-сокеты.

Такой подход к реализации системы тестирования имеет ряд сложностей. Во-первых, компиляция и работоспособность модуля будут отличаться при разных версиях ядра ОС. Во-вторых, при текущей реализации системы тестирования один модуль ядра способен выполнять мониторинг только одного процесса. В-третьих, ошибки, допущенные при написании кода модуля ядра, могут негативно влиять на работоспособность ОС в целом.

Поэтому в качестве альтернативы модулю ядра в системе тестирования авторами был разработан монитор системных вызовов на основе технологии eBPF [22]. Эта технология позволяет запускать специализированные программы в привилегированном контексте, то есть на уровне ядра ОС. Данные программы активизируются при возникновении некоторых событий. Например, в контексте сбора трасс системных вызовов источниками таких событий могут выступать механизмы Kprobes, Kretprobes или Kernel Tracepoints.

Технология eBPF представляет собой виртуальную машину, имеющую 10 регистров: R0-R9, и отдельный указатель на список фреймов стека R10, доступный только для чтения программами eBPF. При этом каждая программа eBPF имеет свой независимый стек в 512 байт. Реализация виртуальной машины eBPF в ядре ОС включает в себя как интерпретатор, так и JIT-компилятор специальных машинных инструкций eBPF в инструкции для выполнения непосредственно на процессоре. Все машинные инструкции eBPF проходят проверку в его верификаторе, представляющем собой встроенный в ядро ОС статический анализатор кода, который предотвращает исполнение программ с небезопасными операциями, такими как небезопасные обращения к памяти или циклы с неограниченным числом итераций. Архитектура среды исполнения eBPF представлена на рис. 2.

Для коммуникации между программами eBPF и для передачи их данных в пользовательское пространство используются так называемые карты eBPF. Они представляют собой общие для пространства ядра и пользовательского пространства ОС реализации структур данных. С использованием этих карт, передаваемых в пользовательское пространство в виде файловых дескрипторов, может осуществляться сохранение данных программ eBPF. Например, массив произвольного размера (карта BPF_MAP_TYPE_ARRAY) позволяет обходить ограничения по размеру стека в 512 байт с целью хранения длинных строк, а карта BPF_MAP_TYPE_RINGBUF – использовать для передачи данных в пользовательское пространство кольцевой буфер. Также может быть использована хэш-таблица (карта BPF_MAP_TYPE_HASH), например, в качестве промежуточного буфера для хранения аргументов системных вызовов конкретных процессов или для осуществления фильтрации процессов по их идентификаторам.

Для управления программами eBPF в ядре ОС предусмотрен специальный системный вызов bpf(), позволяющий взаимодействовать (создавать, удалять, находить элементы) с соответствующими картами и загружать программы. При этом сами программы разрабатываются на значительно ограниченном из-за верификатора варианте языка Си. Так они не могут вызывать произвольные функции ядра ОС. Вместо этого предусмотрен ограниченный состав доступных «вспомогательных» функций [32]. Помимо невозможности выполнения произвольных функций ядра и ограничений по размеру стека верификатором eBPF дополнительно проверяются следующие ограничения: во-первых, все вызываемые в программе eBPF функции должны быть встраиваемыми (inline), во-вторых, возможно использование только циклов с ограниченным числом итераций.

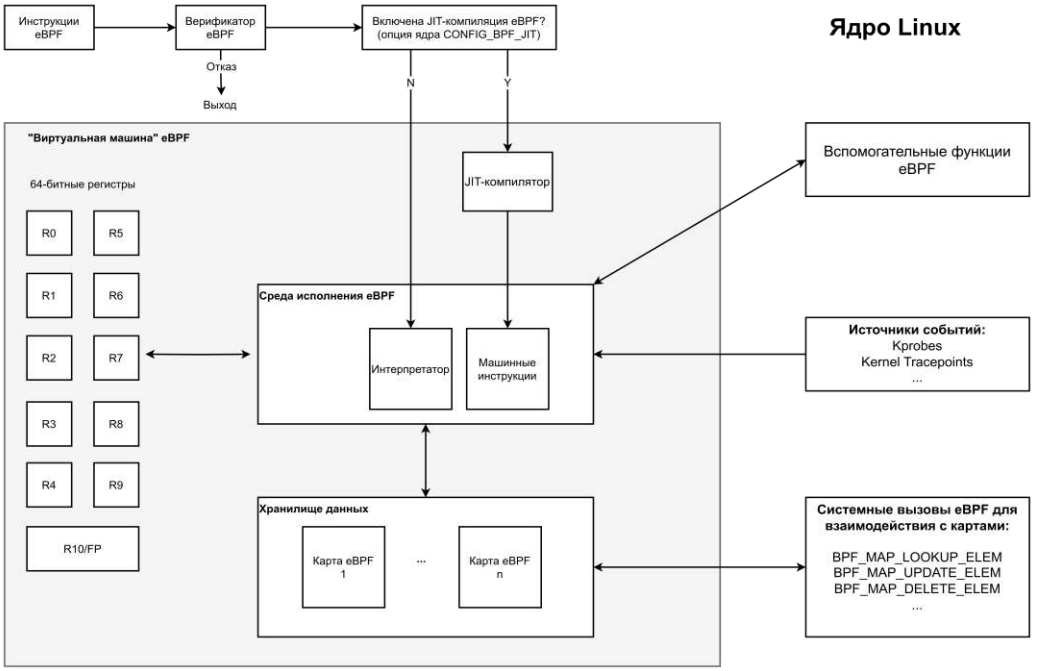


Рис. 2. Архитектура среды исполнения eBPF.
Fig. 2. eBPF runtime architecture.

Одним из ключевых преимуществ технологии eBPF является возможность создавать программы, не требующие компиляции для работы с разными версиями ядра ОС (это делается в соответствии с концепцией Compile Once – Run Everywhere [33]). Такой результат достигается за счет не «прямого» использования внутренних структур ядра ОС, а путем применения специального файла с метаинформацией формата BTF (BPF Type Format), в котором определены сигнатуры вызываемых функций, а также имена и данные о смещении полей структур. Однако в свою очередь для этого требуется ядро ОС с соответствующими активными параметрами компиляции: CONFIG_DEBUG_INFO_BTF и CONFIG_DEBUG_INFO_BTF_MODULES. Эти параметры установлены в отладочных ядрах семейства debug ОС Astra Linux, начиная с релиза 2024 г. (1.8.1), и на стандартных ее ядрах generic, начиная с релиза 2025 года (1.8.2).

В системе тестирования при загрузке программ с помощью библиотеки libbpf, в свою очередь осуществляющей системный вызов bpf(), верификатором проводится их статический анализ, и в случае успеха осуществляется назначение им обработчиков в ядре ОС, то есть инициализация соответствующего механизма перехвата событий. Далее при возникновении события перехвата происходит вызов программы eBPF, который сопровождается выполнением последовательности действий, представленных диаграммой на рис. 3. При запуске самих тестов в системе тестирования для получения информации о системных вызовах используется механизм точек трассировки ядра (Kernel Tracepoints): raw_syscalls/sys_enter – для трассировки аргументов системных вызовов, raw_syscalls/sys_exit – для получения их кодов возврата и передачи данных в кольцевой буфер для чтения программой пользовательского пространства. При этом для получения данных о значениях аргументов функций подсистемы безопасности PARSEC, таких как уровни целостности и конфиденциальности (доступа) процессов, файлов и каталогов, а также для получения их фактических имен, используются механизмы Kprobes и Kretprobes. Для сохранения данных между программами eBPF используется промежуточный буфер, представляющий собой хэш-таблицу. Таким образом, разработанный монитор системных вызовов имеет возможность записывать трассу всех системных вызовов теста.

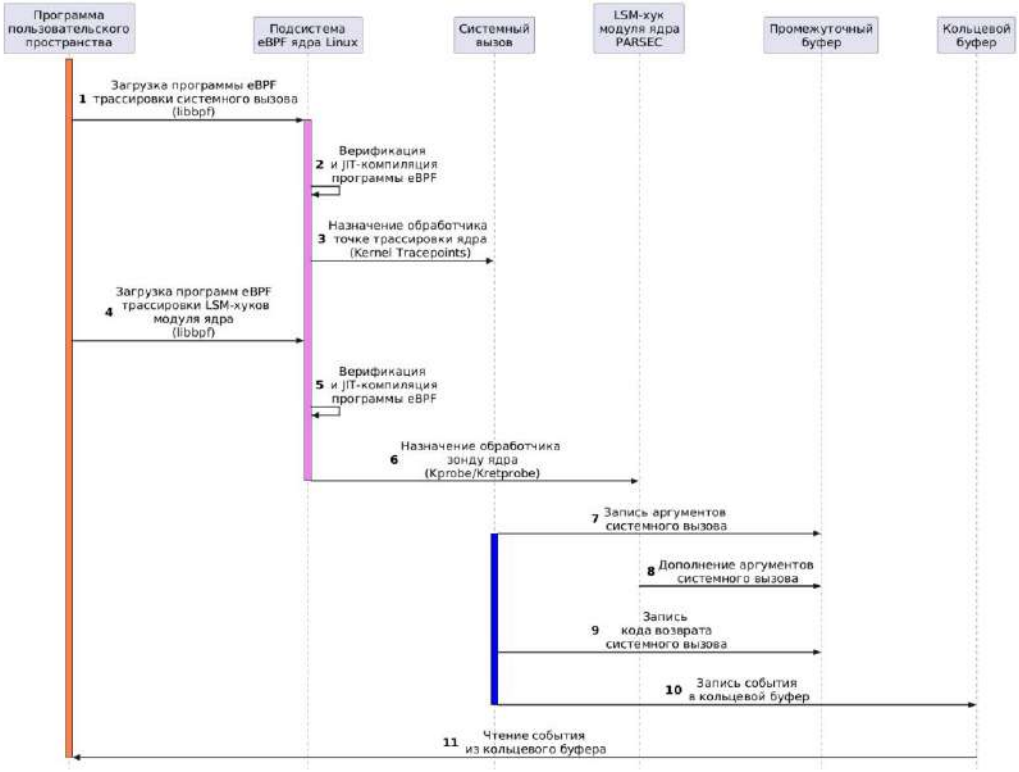


Рис. 3. Диаграмма работы монитора системных вызовов на основе eBPF.
Fig. 3. Sequence diagram of the eBPF-based syscall monitor.

В итоге применение технологии eBPF позволяет расширить функциональные возможности системы тестирования, в том числе за счет распараллеливания тестов.

5. Заключение

В настоящей статье рассмотрены итоги соответствующих рекомендациям ГОСТ Р 59453.4-2025 исследований и разработок по реализации системы тестирование подсистемы безопасности PARSEC ОС Astra Linux на основе нижнеуровневого представления МРОСЛ ДП-модели – PARSEC-модели. Для этого авторами было осуществлено описание PARSEC-модели на языке формального метода Event-B и ее верификация с применением инструментального средства Rodin. Далее на основе результатов [12-13] была разработана адаптированная к подсистеме безопасности PARSEC система тестирования, ориентированная в первую очередь на реализуемые в ОС Astra Linux механизмы МКЦ и МРД. Для повышения эффективности работы этой системы тестирования в статье предложен подход к использованию для ее распараллеливания технологии eBPF.

В дальнейшем планируется продолжить доработку PARSEC-модели с учетом изменений, вносимых как в верхнеуровневое представление МРОСЛ ДП-модели, так и в программный код подсистемы безопасности PARSEC ОС Astra Linux. Для системы тестирования будут разрабатываться новые тесты с целью расширения охвата ими функций механизмов МКЦ и МРД. Кроме того, предполагается исследовать применимость технологии eBPF для разработки системы мониторинга в реальном времени (то есть в штатно работающей ОС по аналогии с технологией Run-Time Verification [34]) подсистемы безопасности PARSEC на соответствие PARSEC-модели.

Список литературы / References

[1]. ГОСТ Р 56939-2024 «Защита информации. Разработка безопасного программного обеспечения. Общие требования». М.: Стандартинформ. 36 с. / GOST R 56939-2024 «Information protection. Secure Software Development. General requirements», 2024. (in Russian).

[2]. ГОСТ Р 59453.1-2021 «Защита информации. Формальная модель управления доступом. Часть 1. Общие положения». М.: Стандартинформ. 16 с. / GOST R 59453.1-2021 «Information protection. Formal access control model. Part 1. General principles», 2021 (in Russian).

[3]. Девянин П.Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками. Учебное пособие для вузов. 3-е изд., перераб. и доп. М.: Горячая линия – Телеком, 2020. 352 с.: ил. / P.N. Devyanin. Security models of computer systems. Control for access and information flows. Hotline-Telecom, 2020, 352 p. (in Russian).

[4]. Выписка из Требований по безопасности информации, утвержденных приказом ФСТЭК России от 2 июня 2020 г. N 76. Доступно по ссылке: https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/trebovaniya-po-bezopasnosti-informatsii-utverzhdeny-prikazom-fstek-rossii-ot-2-iyunya-2020-g-n-76, 05.06.2025 / Excerpts from Requirements for information security approved by FSTEK Russia order #76 of 2nd June 2020. Available at https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/trebovaniya-po-bezopasnosti-informatsii-utverzhdeny-prikazom-fstek-rossii-ot-2-iyunya-2020-g-n-76, accessed 05.06.2025. (in Russian).

[5]. Операционная система специального назначения Astra Linux Special Edition. Доступно по ссылке: https://astra.ru/software-services/os/, 05.06.2025. / Astra Linux Special Edition operating system. Available at https://astra.ru/software-services/os/, accessed 05.06.2025.

[6]. Девянин П.Н., Тележников В.Ю., Третьяков С.В. Основы безопасности операционной системы Astra Linux Special Edition. Управление доступом. Учебное пособие. М., Горячая линия – Телеком, 2022, 148 с. / Devyanin P.N., Telezhnikov V.Y., Tret'yakov S.V. Astra Linux Special Edition security basics. Access control. Hotline-Telecom, 2022, 148 p. (in Russian).

[7]. Bishop M. Computer Security: Art and Science, 2nd edition. Pearson Education Inc., 2018, 1440 p.

[8]. Abrial J.-R., Hallerstede S. Refinement, decomposition, and instantiation of discrete models: Application to Event-B // Fundamenta Informaticae, Volume 77, Issue 1-2, 2007, pp. 1-28.

[9]. Abrial J.-R. Modeling in Event-B: System and Software Engineering. Cambridge University Press, 2010, 612 p.

[10]. Девянин П.Н., Леонова М.А. Приемы по доработке описания модели управления доступом ОССН Astra Linux Special Edition на формализованном языке метода Event-B для обеспечения ее автоматизированной верификации с применением инструментов Rodin и ProB // Прикладная дискретная математика. 2021. № 52. С. 83-96. / P. N. Devyanin, M. A. Leonova, "The techniques of formalization of OS Astra Linux Special Edition access control model using Event-B formal method for verification using Rodin and ProB", Prikl. Diskr. Mat., 2021, no. 52, pp. 83–96 (In Russian).

[11]. ГОСТ Р 59453.4-2025 «Защита информации. Формальная модель управления доступом. Часть 4. Рекомендации по верификации средства защиты информации, реализующего политики управления доступом, на основе формализованных описаний модели управления доступом». М.: Стандартинформ. 20 с. / GOST R 59453.4-2025 «Information protection. Formal model of access control. Part 4. Recommendations on verification of information protection tool implementing access control policies based on formalized descriptions of access control model», 2025 (in Russian).

[12]. Ефремов Д.В., Копач В.В., Корныхин Е.В., Кулямин В.В., Петренко А.К., Хорошилов А.В., Щепетков И.В. Мониторинг и тестирование модулей операционных систем на основе абстрактных моделей поведения системы // Труды ИСП РАН, том 33, вып. 6, 2021, С. 15-26 / Efremov D.D., Kopach V.V., Kornykhin E.V., Kuliamin V.V., Petrenko A.K., Khoroshilov A.V., Shchepetkov I.V. Runtime Verification of Operating Systems Based on Abstract Models. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 6, 2021, pp. 15-26 (in Russian).

[13]. Петренко А.К., Девянин П.Н., Ефремов Д.В., Карнов А.А., Корныхин Е.В., Кулямин В.В., Хорошилов А.В. Методы динамической верификации промышленных средств защиты информации на основе формальных моделей управления доступом // Труды ИСП РАН, 2025, Т. 37, вып. 3, С. 277-290 / Petrenko A.K., Devyanin P.N., Efremov D.V., Karnov A.A., Kornykhin E.V., Kuliamin V.V., Khoroshilov A.V. Methods of runtime verification of industrial information security tools based on formal access control models. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 3, 2025. pp. 277-290 (in Russian)

[14]. Девянин П.Н., Хорошилов А.В., Тележников В.Ю. Формирование методологии разработки безопасного системного программного обеспечения на примере операционных систем // Труды ИСП РАН, том 33, вып. 5, 2021, С. 25-40 / Devyanin P.N., Telezhnikov V.Y., Khoroshilov V.V. Building a methodology for secure system software development on the example of operating systems. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 25-40 (in Russian).

[15]. ГОСТ Р 59453.3-2025 «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке». М.: Стандартинформ. 20 с. / GOST R 59453.3-2025 «Information protection. Formal access control model. Part 3. Recommendations on development», 2025 (in Russian).

[16]. Девянин П.Н. О разработке проекта национального стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке» // Труды ИСП РАН, том 36, вып. 3, 2024, С. 63-82 / Devyanin P.N. On the development of the draft standard GOST R “Information protection. Formal access control model. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 3, 2024, pp. 63-82 (in Russian).

[17]. Девянин П.Н. Результаты переработки уровней ролевого управления доступом и мандатного контроля целостности формальной модели управления доступом ОС Astra Linux // Труды ИСП РАН, том 35, вып. 5, 2023, С. 7-22 / Devyanin P.N. The Results of Reworking the Levels of Role-Based Access Control and Mandatory Integrity Control of the Formal Model of Access Control in Astra Linux. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 5, 2023, pp. 7-22 (in Russian).

[18]. ГОСТ Р 59453.2-2021 «Защита информации. Формальная модель управления доступом. Часть 2. Рекомендации по верификация формальной модели управления доступом». М.: Стандартинформ. 12 с. / GOST R 59453.2-2021 «Information protection. Formal access control model. Part 2. Recommendations on verification of formal access control model», 2021 (in Russian).

[19]. Abrial J.-R., Butler M. et al. Rodin: an open toolset for modelling and reasoning in Event-B. International Journal on Software Tools for Technology Transfer, vol. 12, issue 6, 2010, pp. 447-466.

[20]. Leuschel M., Butler M. ProB: A Model Checker for B. Lecture Notes in Computer Science, vol. 2805, 2003, pp. 855-874.

[21]. Ковалев М.Г. Трассировка сетевых пакетов в ядре Linux с использованием eBPF // Труды ИСП РАН, том 32, вып. 3, 2020, С. 71-78 (на английском языке) / Kovalev M.G. Tracing Network Packets in the Linux Kernel using eBPF. Trudy ISP RAN/Proc.ISP RAS, vol. 32, issue 3, 2020. pp. 71-78.

[22]. Gregg B. BPF Performance Tools. Addison-Wesley Professional, 2019, 880 p.

[23]. Tetragon. Overview. Available at https://tetragon.io/docs/overview/, accessed 05.06.2025.

[24]. The Falco Project. Developer Guide. Available at https://falco.org/docs/developer-guide/, accessed 05.06.2025.

[25]. Aqua Security. Aqua Tracee. Available at https://www.aquasec.com/products/tracee/, accessed 05.06.2025.

[26]. Wright C., Cowan C., Smalley S., Morris J., Kroah-Hartman G. Linux Security Modules: General Security Support for the Linux Kernel. Proc. of the 11-th USENIX Security Symposium, pp. 17–31, 2002.

[27]. Девянин П.Н., Старостин А.А., Панов Д.С., Усачев С.В. Проектирование и развитие механизма мандатного контроля целостности в операционной системе Astra Linux // Труды ИСП РАН, том 37, вып. 2, 2025, С. 61-78 / Devyanin P.N., Starostin A.A., Panov D.S., Usachev S.V. Design and Development of a Mandatory Integrity Control Mechanism in the Astra Linux Operating System. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 2, 2025, pp. 61–78 (in Russian).

[28]. D. Efremov and I. Shchepetkov. Runtime Verification of Linux Kernel Security Module. Proc. Of International Workshop on Formal Methods, LNCS 12233:185-199, Springer, 2020.

[29]. Allure Report Documentation. Available at https://allurereport.org/docs/, accessed 05.06.2025.

[30]. Карнов А.А. Проблема неопределенности в анализе трасс на основе высокоуровневых моделей в контексте динамической верификации // Труды ИСП РАН, том 36, вып. 4, 2024, С. 169-182 / Karnov A.A. Uncertainty Problem in High-Level Model-Based Trace Analysis as Part of Runtime Verification. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 4, 2024, pp. 169-182 (In Russian).

[31]. pytest: helps you write better programs. Available at https://docs.pytest.org/en/stable/, accessed 05.06.2025.

[32]. bpf-helpers(7) – Linux manual pages. Available at https://man7.org/linux/man-pages/man7/bpf-helpers.7.html, accessed 05.06.2025.

[33]. BPF CO-RE, eBPF Docs. Available at https://docs.ebpf.io/concepts/core/, accessed: 05.06.2025.

[34]. Linux 6.0 Adding Run-Time Verification For Running On Safety Critical Systems. Available at https://www.phoronix.com/news/Linux-6.0-Runtime-Verification, accessed 05.06.2025.

Информация об авторах / Information about authors

Петр Николаевич ДЕВЯНИН – член-корреспондент Академии криптографии России, доктор технических наук, профессор, научный руководитель ООО «РусБИТех-Астра» («Группа Астра»). Область интересов: теория информационной безопасности, формальные модели безопасности компьютерных систем, разработка безопасного программного обеспечения, операционные системы семейства Linux.

Petr Nikolaevich DEVYANIN – Doctor of Technical Sciences, corresponding member of Russian Academy of Cryptography, professor, scientific director in RusBITech-Astra (Astra Linux). Field of Interest: information security theory, formal security models of computer systems, secure software development, operating systems of Linux family.

Сергей Сергеевич ЖИЛЯКОВ – инженер ООО «РусБИТех-Астра» («Группа Астра»). Область интересов: формальные модели безопасности компьютерных систем, операционные системы семейства Linux.

Sergey Sergeevich ZHILIAKOV – engineer in RusBITech-Astra (Astra Linux). Field of Interest: formal security models of computer systems, operating systems of Linux family.

Александр Игоревич СМИРНОВ – инженер ООО «РусБИТех-Астра» («Группа Астра»). Область интересов: формальные модели безопасности компьютерных систем, операционные системы семейства Linux, разработка безопасного программного обеспечения.

Alexander Igorevich SMIRNOV – engineer in RusBITech-Astra (Astra Linux). Field of Interest: formal security models of computer systems, operating systems of Linux family, secure software development.

DOI: 10.15514/ISPRAS-2025-37(6)-18

Статический анализ языка Visual Basic .NET

[1,2] *Карцев В. С., ORCID: 0000-0001-7482-0835 <karcev.vs@ispras.ru>*
[1,3] *Игнатьев В. Н., ORCID: 0000-0003-3192-1390 <valery.ignatyev@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.*

[2] *Физтех-школа радиотехники и компьютерных технологий МФТИ, 141701, Россия, г. Долгопрудный, ул. Первомайская, д. 5.*

[3] *Факультет вычислительной математики и кибернетики МГУ им. М.В. Ломоносова, 119991, Россия, г. Москва, ул. Колмогорова, д. 1, стр. 52.*

Аннотация. В работе представлена реализация статического анализа для языка Visual Basic .NET в рамках промышленного инструмента SharpChecker. С помощью фреймворка компилятора Roslyn в SharpChecker была интегрирована поддержка языка Visual Basic .NET. Это позволило выполнять статический анализ исходного кода на языке Visual Basic .NET. В рамках работы также был создан репрезентативный набор синтетических тестов, содержащий суммарно более 2000 тестов. Тестирование производилось как на созданной выборке тестов, так и на наборе реальных проектов с открытым исходным кодом суммарным объемом более 1.6 млн. строк кода. Было обнаружено 7926 новых предупреждений в исходном коде на языке Visual Basic .NET, из которых 1093 были проанализированы и размечены вручную. Итоговая точность анализа составила 84.72%. Кроме того, были обнаружены предупреждения, связанные с кодом на языках C# и Visual Basic .NET одновременно, что показало возможность производить межъязыковой анализ в проектах, которые содержат сразу два языка платформы .NET. Добавление поддержки языка Visual Basic .NET в инструмент SharpChecker не отразилось на времени работы и на качестве анализа для языка C#.

Ключевые слова: статический анализ; Visual Basic .NET; символьное исполнение; анализ потоков данных; генерация кода.

Для цитирования: Карцев В. С., Игнатьев В. Н. Статический анализ языка Visual Basic .NET. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 37–52. DOI: 10.15514/ISPRAS–2025–37(6)–18.

Static Analysis of Visual Basic .NET Language

[1,2] *Karcev V. S., ORCID: 0000-0001-7482-0835 <karcev.vs@ispras.ru>*
[1,3] *Ignatiev V. N., ORCID: 0000-0003-3192-1390 <valery.ignatyev@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*

[2] *Departament of Radio Engineering and Cybernetics MIPT, 5, Pervomayskaya Str., Dolgoprudny, 141701, Russia.*

[3] *CMC Faculty Lomonosov State University, 1c52, Kolmogorov Str., Moscow, 119991, Russia.*

Abstract. The paper presents the implementation of static analysis for the Visual Basic .NET language within the industrial tool SharpChecker. Using the Roslyn compiler framework, support for the Visual Basic .NET language was integrated into SharpChecker, enabling static analysis of Visual Basic .NET source code. As part of this work, a representative set of synthetic tests was created, comprising over 2000 test cases. Testing was conducted both on this synthetic dataset and on a collection of real-world open-source projects totaling more than 1.6 million lines of code. A total of 7926 new warnings were detected in Visual Basic .NET source code, of which 1093 were manually reviewed and labeled. The final analysis accuracy reached 84.72%. Additionally, warnings related to code written in both C# and Visual Basic .NET were discovered, demonstrating the feasibility of cross-language analysis in projects that include both .NET platform languages. It was also found that adding Visual Basic .NET language support to SharpChecker had no impact on the performance or the quality of analysis for the C# language.

Keywords: static analysis; Visual Basic .NET; symbolic execution; dataflow analysis; code generation

For citation: Karcev V. S., Ignatiev V. N. Static analysis of Visual Basic .NET language. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 37-52 (in Russian). DOI: 10.15514/ISPRAS-2025-37(6)-18.

1. Введение

Visual Basic .NET – это интерпретируемый, объектно-ориентированный язык со статической типизацией, разработанный корпорацией Microsoft. Visual Basic .NET произошёл от языка Visual Basic (сейчас известен как Visual Basic 6.0 или VB6), также разработанного компанией Microsoft. Несмотря на то, что Visual Basic .NET – это очередное, хоть и значительное, обновление языка Visual Basic, обратная совместимость с VB6 сохранена не была [1]. Ключевой особенностью языка является его ориентированность на среду исполнения .NET при высокой простоте перехода с VB6. Для упрощения перехода на платформу .NET компанией Microsoft была разработана утилита Migration Wizard, которая позволяет конвертировать проекты с VB6 на Visual Basic .NET.

Visual Basic .NET является восьмым по популярности языком согласно рейтингу TIOBE [2]. Он широко используется в корпорациях и государственных организациях, что объясняется существованием большой устоявшейся кодовой базы на Visual Basic .NET и VB6 внутри компаний. При этом, переход с устаревшего VB6 на Visual Basic .NET может повлечь большое количество ошибок в существующих проектах, так как утилита Migration Wizard не гарантирует корректности и конвертированный код требует ручной доработки [3].

Также стоит отметить, что в некоторых современных проектах, ориентированных на платформу .NET прослеживается тенденция на использование одновременно нескольких языков платформы .NET. В таких проектах игнорирование исходного кода на языке Visual Basic .NET может привести к пропуску ошибок, связанных с этим кодом. В качестве примера такого проекта можно рассмотреть компилятор для языков C# и Visual Basic .NET Roslyn [4], включающий 1.52 млн. строк кода на Visual Basic .NET. Для того чтобы считать такое ПО безопасным, согласно ГОСТ Р 56939-2024 о разработке безопасного ПО, необходимо производить, в том числе, контроль качества исходного кода [5]. То есть, для соблюдения

требований ГОСТ, необходимо производить статический анализ не только C#, но и Visual Basic .NET.

Для оценки сложности интеграции и возможного качества результатов авторами ранее был реализован прототип поддержки языка Visual Basic .NET в промышленном инструменте SharpChecker. Прототипирование показало, что без кардинальных изменений анализатора можно достигнуть сравнимых с C# показателей полноты и точности [6]. При этом появилась возможность производить статический анализ всех типов – как синтаксический анализ, так и более сложные, чувствительные к контексту и путям исполнения типы анализа. Это показало актуальность доработки прототипа до полноценного решения, которое может быть предоставлено конечным пользователям.

Статический анализ исходного кода на языке Visual Basic .NET был реализован в рамках промышленного статического анализатора SharpChecker, разработанного в ИСП РАН [7]. SharpChecker предусматривает следующий конвейер анализа:

1. перехват сборки и сбор информации для индексации исходного кода;
2. повторная сборка и анализ, разделенный на следующие этапы:
   a. синтаксический анализ;
   b. анализ потоков данных;
   c. чувствительный к путям выполнения межпроцедурный анализ на основе символьного выполнения;
   d. межпроцедурный, контекстно-чувствительный анализ помеченных данных [8];
3. отображение обнаруженных предупреждений в графическом пользовательском интерфейсе с возможностью разметки и навигации по исходному коду.

Для проверки работоспособности и отладки инструмента SharpChecker используются синтетические тесты, представляющие собой корректные компилируемые программы с допущенными и размеченными ошибками.

Помимо анализа исходного кода, SharpChecker собирает метрики исходного кода – численные значения, отражающие его характеристики. В качестве примера метрик можно привести цикломатическую сложность, связность и другие [9]. Данные метрики могут быть использованы для дальнейшего анализа качества кода или автоматической разметки сгенерированных предупреждений [10].

Архитектура SharpChecker подразумевает наличие отдельных модулей (далее детекторов), производящих поиск определенных типов дефектов или производящих сбор дополнительных данных, например, информации о значениях полей [11].

SharpChecker был выбран в качестве платформы для реализации статического анализа Visual Basic .NET, так как он использует для сборки и построения промежуточных представлений исходного кода компилятор для C# и Visual Basic .NET Roslyn [4]. Это позволяет производить межъязыковой анализ для проектов, использующих оба эти языка.

Целью данной работы является разработка методов и алгоритмов для поддержки статического анализа кода на Visual Basic .NET в инструменте SharpChecker.

Для достижения поставленной цели необходимо:

- разработать механизмы поддержки языка Visual Basic .NET на всех этапах анализа;
- создать репрезентативный набор тестов;
- обеспечить навигацию по исходному коду в интерфейсе пользователя;
- разработать подсистему сбора метрик исходного кода для языка Visual Basic .NET;
- произвести тестирование реализованных методов и механизмов на наборе проектов с открытым исходным кодом.

2. Существующие решения

Большинство промышленных статических анализаторов не поддерживают язык Visual Basic .NET [12]. Например, Klocwork [13] и PVS-Studio не поддерживают Visual Basic .NET, несмотря на то, что они поддерживают C# и платформу .NET.

Тем не менее, некоторые статические анализаторы реализуют анализ языка Visual Basic .NET. Например, Visual Basic .NET поддерживается в инструменте ReSharper [14-15]. ReSharper направлен в первую очередь на быстрый анализ кода на языках C# и Visual Basic .NET для обеспечения помощи разработчику непосредственно при написании кода. Поиск большинства типов ошибок в ReSharper производится в рамках одного файла. В рамках всего проекта возможен только поиск ошибок, связанных с использованием и доступом к членам объявленных типов. Таким образом, при анализе ReSharper не учитывает содержимое других файлов, что приводит к невозможности обнаружить нетривиальные ошибки, для поиска которых необходим трудоемкий межпроцедурный и межмодульный анализ.

Статический анализ языка Visual Basic .NET также представлен в инструменте SonarQube [16-17]. Этот инструмент используется для обнаружения ошибок, уязвимостей и подозрительных мест в коде на основе правил. Он поддерживает 21 язык, среди которых присутствует и Visual Basic .NET. Отметим, что количество правил для языка Visual Basic .NET значительно меньше, чем для других языков. Большинство правил направлены на поиск простых ошибок, таких как поиск классов, объявленных вне пространства имен или использование слабых криптографических алгоритмов. Однако существуют и более сложные правила, например, отсутствие снятия блокировки на одном или нескольких путях исполнения или разыменование `Nothing` [18].

Инструмент поиска уязвимостей Kiuwan также поддерживает анализ языка Visual Basic .NET. Kiuwan предназначен для обнаружения уязвимостей и оценки качества исходного кода. Kiuwan обнаруживает, в том числе, переполнения буфера, инъекции команд, межсайтовый скриптинг, SQL-инъекции и другие типы уязвимостей [19]. Однако данный инструмент не предназначен для обнаружения других типов ошибок, таких, как недостижимый код или разыменование `null` (или `Nothing` в случае Visual Basic .NET).

3. Интеграция языка Visual Basic .NET

3.1 Генерация тестовой выборки

Для отладки и проверки инструмента SharpChecker в процессе добавления поддержки Visual Basic .NET необходим набор тестов. В SharpChecker уже существует репрезентативный набор тестов, созданный более, чем за 5 лет для языка C#. По этой причине было принято решение создать инструмент автоматической трансляции набора тестов с C# на Visual Basic .NET для создания аналогичного набора тестов для нового языка с минимальными затратами по времени. В рамках статьи [6] был описан инструмент автоматической генерации набора синтетических тестов. Данный инструмент состоит из двух компонентов: механизма импорта и экспорта тестовой выборки и механизма трансляции кода с языка C# на язык Visual Basic .NET.

Для импорта всех тестов в инструмент трансляции в первую очередь необходимо обнаружить все файлы, содержащие тесты. Далее для каждого файла, в котором были обнаружены тесты, с помощью Roslyn создается абстрактное синтаксическое дерево и производится поиск функций, помеченных атрибутом `[Test]`. В таких функциях исходный код программы с размеченными ошибками передается в функцию `VerifyCSharp`, производящую анализ этого исходного кода и сверяющего результаты анализа с ручной разметкой. Данный исходный код извлекается из вызова проверочной функции и передается на трансляцию из C# на Visual Basic .NET.

В первую очередь, перед трансляцией производится переименование символов, объявленных в коде, с помощью их нумерации. Далее для исходного кода строится синтаксическое дерево и создается эквивалентное из аналогичных узлов для Visual Basic .NET. После перевода всех тестов транслятор воссоздает структуру исходного синтаксического дерева файла с тестами, заменяя проверочные методы для C# на аналогичные для Visual Basic .NET и подставляя в качестве параметра переведенный исходный код.

Из 2680 тестов для языка C#, содержащихся в SharpChecker, на язык Visual Basic .NET удалось автоматически перевести 1928 тестов. Тесты, которые не удалось перевести, используют функциональность C#, которая не имеет аналогий в Visual Basic .NET. Таким образом, в результате использования инструмента автоматического перевода тестов удалось получить репрезентативный набор тестов для языка Visual Basic .NET, практически эквивалентный аналогичному для языка C#.

Заметим, что данный инструмент производит перевод на основе синтаксического дерева, заменяя конструкции языка C# соответствующими для языка Visual Basic .NET. Благодаря этому, переведенный исходный код после преобразования во внутреннее представление компилятора Roslyn, будет трактоваться эквивалентно аналогичному на языке C#. Таким образом, данный метод позволяет гарантировать отсутствие различий в семантике исходного кода.

3.2 Адаптация существующих детекторов

Анализ исходного кода в инструменте SharpChecker происходит с использованием различных методов анализа. В первую очередь стоит отметить детекторы, производящие синтаксический анализ. На данном этапе, помимо абстрактного синтаксического дерева, доступны также таблица символов и дерево операций [20]. Операции, предоставляемые платформой Roslyn, являются универсальным представлением различных паттернов исходного кода, общих для всех языков. Так, например, существуют операции, обозначающие в обобщенном виде циклы, ветвления, методы, классы и так далее. Однако существует также множество специфичных для языка конструкций, которые не имеют аналогов в виде операций.

В основном синтаксический анализ использует обработку либо узлов АСД, либо символов, либо операций, либо их комбинации. В случае, если детектор использует только операции и/или символы, то его доработка необходима, только если существует некая специфика, присущая языку Visual Basic .NET. В остальных случаях такие детекторы способны обрабатывать код на любом из языков. В случае, если детектор использует обработку синтаксических узлов (например, если аналогичных им операций не существует), необходимо добавить поддержку аналогичных синтаксических узлов для Visual Basic .NET.

После синтаксического анализа следует этап символьного исполнения. Детекторы на этапе символьного исполнения имеют сложный граф зависимостей. Часть детекторов используется для предварительного анализа исходного кода и сбора данных о нем. Назовем набор таких детекторов «ядром». Обычные детекторы, в свою очередь, используют собранную им информацию. Таким образом, в первую очередь необходимо реализовать поддержку языка Visual Basic .NET именно в ядре. Это поможет инструменту SharpChecker собирать информацию об исходном коде на языке Visual Basic .NET и составлять резюме методов для дальнейшего анализа. Для устранения ошибок и неточностей в анализе отдельные детекторы также требуют доработки.

Детекторы, использующие анализ помеченных данных, работают несколько иначе. Каждый детектор на данном этапе представлен отдельным правилом, описанным в виде JSON-файла. Движок анализа помеченных данных использует эти правила для определения характеристик стока и истока помеченных данных, а также пути распространения этих данных. Таким образом, сами правила не нуждаются в доработке, за исключением добавления новых стоков,

истоков или распространителей, специфичных только для языка Visual Basic .NET. Доработка необходима только основному движку.

По большей части, некорректная работа инструмента SharpChecker с исходным кодом на языке Visual Basic .NET связана с тем, что Visual Basic .NET и C# имеют разные типы синтаксических узлов. В результате детекторы, реализующие анализ узлов АСД, игнорируют узлы АСД для кода, написанного на Visual Basic .NET. Помимо этого, язык Visual Basic .NET имеет ряд отличий от языка C#:

- встроенный XML-синтаксис;
- инициализация полей вызовом конструктора с помощью оператора `As`;
- сравнение объектов по ссылке с помощью отдельных операторов `Is` и `IsNot`;
- дополнительный метод `IsNothing` для сравнения объектов ссылочного типа со значением `Nothing`;
- оператор `ReDim` для реаллокации массива;
- иное представление цепочки ветвлений в АСД (вся цепочка в виде списка вместо вложенных узлов);

3.2.1 Синтаксический анализ

Основными особенностями детекторов, производящих синтаксический анализ, являются высокая скорость анализа, низкое потребление ресурсов и независимость от других компонентов инструмента SharpChecker. Это приводит к необходимости производить доработку каждого детектора по отдельности до достижения приемлемого качества анализа.

Помимо синтаксического дерева, Roslyn предоставляет универсальные представления исходного кода, такие как дерево операций или таблица символов. В инструменте SharpChecker присутствуют детекторы, которые изначально были реализованы с использованием только анализа дерева операций или таблицы символов. К таким детекторам можно отнести, например, `UselessCall` или `DuplicateEnumMember`. Однако большинство детекторов используют анализ узлов АСД, что приводит к необходимости их доработки. Для различных детекторов были использованы различные подходы к адаптации:

- поиск неиспользуемых полей и забытых модификаторов `ReadOnly` – добавлена поддержка XML-синтаксиса, добавлена поддержка синтаксических узлов Visual Basic .NET;
- детектор пустых интерфейсов и детектор идентичных методов – добавлена обработка синтаксических узлов Visual Basic .NET;
- поиск перекрытия имен символов – анализ узлов АСД был заменен анализом операций и символов;
- обнаружение скопированных участков кода с некорректными изменениями – добавлен анализ узлов АСД для Visual Basic .NET при анализе компиляции;
- поиск некорректных сравнений чисел с плавающей точкой и сравнений чисел с плавающей точкой с целыми, поиск доступа к переменным с некорректными блокировками на основе статистики, сравнений объектов не ссылочного типа по ссылке, детектор использования слабых криптографических методов, поиск сравнений объектов не ссылочного типа с `null` (`Nothing`), поиск некорректных использований атрибута `[ThreadStatic]`, неверных использований форматных строк, присваиваний переменных или полей самих в себя, идентичных возвращаемых значений на разных путях исполнения, детектор отсутствия вызова базового метода в наследнике, поиск явно заданных математических констант и конструкций `switch - case`, не рассматривающих все члены перечисления – переписаны с использованием операций.

3.2.2 Анализ потоков данных

На этапе анализа потоков данных реализован только один детектор – `UnusedValue`. Он позволяет обнаруживать значения, не использованные до момента выхода из области видимости или до замещения другим значением.

При анализе он использует только обработку операций, что позволяет ему обнаруживать ошибки без значительных доработок. Однако, он не поддерживает обработку встроенного в Visual Basic .NET синтаксиса XML. Таким образом, для удаления ложных предупреждений о неиспользуемых значениях необходимо добавить поддержку данного вида синтаксиса. Кроме того, в Visual Basic .NET невозможно производить перехват исключений определенного типа без присвоения их в переменную. Таким образом, предупреждения о неиспользованных значениях перехваченных исключений являются заведомо ложными, и их необходимо подавлять при анализе.

3.2.3 Символьное исполнение

На этапе символьного исполнения производится межпроцедурный, чувствительный к путям и контексту выполнения анализ, позволяющий обнаруживать сложные ошибки [21]. Из-за использования эвристик, сокращающих время анализа сложных конструкций без значительных потерь в качестве анализа, этот тип анализа допускает ложные предупреждения и пропущенные ошибки [22].

Для реализации поддержки анализа языка Visual Basic .NET необходимо в первую очередь адаптировать ядро. Для этого следует добавить обработку узлов АСД для языка Visual Basic .NET, а также реализовать анализ специфичной для Visual Basic .NET функциональности. В частности, в рамках ядра необходимо анализировать XML-литералы, инициализации с помощью оператора As, использования специфичных для Visual Basic .NET методов, таких как `IsNothing` и использования операторов `ReDim`.

Для некоторых детекторов на данном этапе требуются доработки для подавления ложных предупреждений. В частности, в отдельных детекторах производится анализ синтаксического дерева. Например, в связи с отличием структуры дерева, был доработан поиск операторов `Return` внутри ветвления в детекторе недостижимого кода для кода на языке Visual Basic .NET.

3.3 Генерация дополнительных артефактов

3.3.1 Индексация исходного кода

Индексация исходного кода – процесс сбора информации о расположениях всех объявлений, определений и использований символов, необходимый для реализации навигации по исходному коду. В рамках SharpChecker индексатор исходного кода анализирует абстрактное синтаксическое дерево проекта, собирая информацию об объявлениях, определениях и использованиях символов в исходном коде. Для индексации по коду на языке Visual Basic .NET необходимо добавить обработку специфичных для Visual Basic .NET узлов АСД. Для индексации исходного кода производится синтаксический анализ, в рамках которого обрабатываются узлы, соответствующие объявлениям, определениям и использованиям символов.

Для поддержки индексации исходного кода на языке Visual Basic .NET был реализован синтаксический анализатор, аналогичный существующему для C#. В нем были реализованы методы обработки синтаксических узлов Visual Basic .NET. При анализе проекта инструмент SharpChecker выбирает подходящий индексатор исходного кода, исходя из языка, на котором реализован анализируемый код. Кроме того, для проектов, использующих сразу оба языка

(Visual Basic .NET и C#), поддерживается индексация каждого из языков по отдельности соответствующим индексатором с последующим объединением собранных данных.

Индексатор исходного кода экспортирует данные в едином формате для всех языков. Механизм навигации по исходному коду в пользовательском интерфейсе не нуждается в доработке, так как для навигации не требуется дополнительная информация, помимо собранной индексатором.

3.3.2 Сбор метрик исходного кода

Из-за высокой схожести языков для Visual Basic .NET актуален тот же набор метрик, что и для C#. Таким образом, для получения метрик исходного кода необходимо доработать существующий в SharpChecker механизм сбора метрик для обработки нового языка. Так как сбор метрик, как и индексация исходного кода, использует анализ АСД, необходимо добавить поддержку синтаксических узлов языка Visual Basic .NET.

Механизм сбора метрик реализован на базе механизма `CSharpSyntaxWalker`. Однако в данном случае реализация дублирующего сборщика метрик для Visual Basic .NET оказалась невозможна, так как для сбора метрик необходимо иметь полный граф сущностей для всего анализируемого проекта. В связи с этим возникла необходимость производить сбор метрик в рамках одного механизма, чтобы избежать в дальнейшем сложного слияния данных из двух сборщиков.

Для реализации универсального инструмента сбора метрик был использован механизм `SyntaxWalker` платформы Roslyn. Данный механизм является базовым для `CSharpSyntaxWalker` и `VisualBasicSyntaxWalker`. В отличие от них, он не реализует методов для обработки различных типов узлов и имеет лишь три метода – `Visit` для обработки любого узла синтаксического дерева, `VisitTrivia` для обработки конструкций, не принимающих участия в компиляции исходного кода и `VisitToken` для обработки токенов, на которые был разбит исходный код при построении АСД.

Универсальный механизм сбора метрик переопределяет метод `Visit`, в котором вручную проверяется тип обрабатываемого узла, и, в зависимости от него, выбирается обработчик для этого типа узлов. Большинство узлов в языках Visual Basic .NET и C# обрабатываются идентичным образом, однако в некоторых случаях имеются значительные различия между этими языками. В частности, значительно отличается обработка ветвлений с множеством веток, что обусловлено отсутствием вложенности узлов.

4. Тестирование

Тестирование статического анализа для языка Visual Basic .NET производилось несколькими путями – на сгенерированной тестовой выборке и на наборе проектов с открытым исходным кодом. Кроме того, на основе собранных для Visual Basic .NET метрик исходного кода было произведено предсказание истинности полученных предупреждений.

4.1 Результаты на наборе синтетических тестов

Набор синтетических тестов для языка C# был собран более чем за пять лет на основе истинных и ложных ошибок, обнаруженных в реальных проектах. Таким образом, данный набор покрывает множество известных сценариев возникновения ошибок в коде. Проверка работоспособности статического анализатора на наборе переведенных на Visual Basic .NET тестов может показать приблизительное качество анализа и оценить, насколько оно уступает качеству для языка C#.

Тестовая выборка включает в себя как автоматически переведенные тесты, основанные на аналогичной тестовой выборке для языка C#, так и новые тесты, реализованные для проверки специфичных для Visual Basic .NET ситуаций. Общая тестовая выборка содержит 2038 тестов

и покрывает все поддерживаемые для Visual Basic .NET типы ошибок. Из 2038 тестов 311 тестов были исключены, так как детекторы, проверяемые этими тестами, заведомо не поддерживают язык Visual Basic .NET или же логика теста не поддерживается инструментом SharpChecker. Результаты для поддерживаемых тестов приведены в табл. 1.

Табл. 1. Результаты по отдельным типам анализа.
Table 1. Results for analysis types.

Набор тестов	**Пройдено**	**Не пройдено**	**Доля пройденных**
Синтаксический анализ	106	2	98.1%
Анализ графа вызовов	42	0	100.0%
Анализ потоков данных	49	0	96.0%
Символьное исполнение	1313	71	94.9%
Анализ помеченных данных	95	3	96.9%
Все тесты	1650	76	95.5%

Инструмент SharpChecker успешно прошел 95.5% синтетических тестов, созданных для языка Visual Basic .NET. В то же время, для тестов на языке C# доля пройденных тестов составляет 98.5%. Таким образом, удалось добавить поддержку для большинства сценариев возникновения ошибок, рассмотренных в тестовом наборе.

4.2 Тестирование на реальных проектах

Для тестирования анализа в реальных условиях была собрана выборка проектов с открытым исходным кодом, состоящая из 5 проектов суммарным объемом 1.5 миллиона строк кода на языке Visual Basic .NET. Кроме того, в выборке присутствует и код на языке C#.

Тестирование проводилось на компьютере, оборудованном процессором Intel® Core™ i7-6700 и 32 гигабайтами оперативной памяти. Анализ набора проектов занял 2 часа, 4 минуты и 47 секунд, было сгенерировано 15816 предупреждений, из которых 7926 относятся к исходному коду на языке Visual Basic .NET, а оставшиеся 7890 к коду на C#. Часть полученных предупреждений была проанализирована и размечена вручную. Всего было размечено 1161 предупреждения, относящихся к различным детекторам. Среди них было обнаружено 977 истинных и 184 ложных предупреждений. Разметка производилась для каждого типа предупреждений по отдельности вслепую – для каждого типа случайным образом выбирался набор предупреждений для разметки. Это позволило оценить как общее качество анализа, так и качество отдельно взятых детекторов и типов анализа. Детекторы, имеющие заведомо высокую точность анализа из-за простоты их реализации (например, поиск неиспользуемых полей или забытых модификаторов `ReadOnly`), не подвергались разметке.

Таким образом, доля истинных предупреждений составила 84.2%. При этом для анализа языка C# доля истинных предупреждений составляет 88.7%. Сравнение доли истинных предупреждений для отдельных типов анализа приведено в табл. 2.

Из таблицы видно, что в среднем качество анализа для языка Visual Basic .NET ниже, чем качество для C#, однако достаточно высоко для использования инструмента в промышленных целях. Худшее качество можно объяснить тем, что детекторы были

доработаны таким образом, чтобы не изменить поведения в случае анализа кода C#, что наложило ограничения на обработку конструкций Visual Basic .NET.

Табл. 2. Сравнение качества анализа по отдельным типам анализа.
Table 2. Comparison of the quality of analysis for individual types of analysis

Тип анализа	**Visual Basic .NET Доля истинных, %**	**C# Доля истинных, %**
Синтаксический анализ	100.0%	94.6%
Анализ графа вызовов	98.4%	90.1%
Анализ потоков данных	71.5%	91.9%
Символьное исполнение	76.5%	79.5%
Анализ помеченных данных	83.2%	94.7%

Кроме того, было обнаружено более 200 межъязыковых предупреждений. В таких предупреждениях причина возникновения ошибки и место самой ошибки находятся в исходном коде на разных языках. В частности, были обнаружены предупреждения о разыменовании null, основанные на статистике вызовов метода. После добавления анализа языка Visual Basic .NET, SharpChecker начал обнаруживать вызовы библиотечных методов не только в исходном коде на C#, но и на Visual Basic .NET, что позволило построить более корректную статистику и повысить точность анализа.

Сравнение с другими статическими анализаторами не производилось из-за закрытости последних или непоказательности сравнения. В частности, ReSharper и Kiuwan предназначены для задач, отличных от задач инструмента SharpChecker.

Рассмотрим некоторые из предупреждений, обнаруженных в исходном коде на языке Visual Basic .NET.

4.2.1 UNREACHABLE_CODE

Предупреждение на листинге 1 о недостижимом коде было обнаружено в проекте `EVE-IPH` [23] в файле `ManufacturingFacility.vb` в строке 2026.

```
 1 Public Class ManufacturingFacility
 2     Private Sub LoadFacilities(...)
 3         If C1 Or C2 Or C3 Or C Then
 4             ' CODE
 5         Else
 6             If Not C Then
 7                 ' CODE
 8             Else
 9                 ' UNREACHABLE_CODE
10             End If
11         End If
12     End Sub
13 End Class
```

Листинг 1. Недостижимый код в проекте EVE-IPH.
Listing 1. Unreachable code in EVE-IPH project.

В данном примере `C1`, `C2`, `C3` и `C` – некоторые условия, не значимые в рамках рассматриваемого примера. В случае, если условие, проверяемое в строке 3, истинно, исполнение не может дойти до строки 9. В противном случае, если это условие ложно, то можно сделать вывод, что каждая из частей условия `C1`, `C2`, `C3` и `C` – ложна. Таким образом, в случае входа в ветку `Else` в строке 5 условие `Not C` будет истинно всегда. Отсюда можно сделать вывод, что код, расположенный в ветке `Else` в строке 8 будет недостижим при любых путях исполнения программы.

4.2.2 DEREF_AFTER_NULL

Предупреждение о разыменовании объекта после проверки его на равенство `Nothing`, приведенное на листинге 2, было обнаружено в компиляторе `Roslyn` версии 3.2.0, в строке 1909 файла `Binder_Lookup.vb`.

```
1 If cont IsNot Nothing And cont.SpecialType = ... Then
2     Return
3 End If
```

Листинг 2. Разыменование Nothing в проекте Roslyn.
Listing 2. Nothing dereference in Roslyn project.

В данном примере производится сравнение объекта `cont` со значением `Nothing`. Логика данного сравнения предполагает, что вторая часть выражения будет вычислена только при истинности первой части выражения. Однако оператор `And` в Visual Basic .NET не ленивый и безусловно вычисляет оба своих операнда. Таким образом, вне зависимости от результата сравнения произойдет разыменование `cont`. В данном случае следует использовать ленивый оператор `AndAlso` для предотвращения возможной ошибки.

4.3 Быстродействие

Для сравнения быстродействия использовался проект с открытым исходным кодом Roslyn – платформа для компиляции и анализа исходного кода на языках C# и Visual Basic .NET [4]. Данный проект содержит 2.2 млн. строк кода на языке C# и 1.5 млн. строк кода на языке Visual Basic .NET.

Для сравнения был произведен анализ проекта с отключенным и с включенным анализом Visual Basic .NET. Результаты тестирования приведены в табл. 3.

Табл. 3. Корреляция объема кода и времени анализа.
Table 3. Correlation between code size and analysis time.

Параметры анализа	Количество строк	Время анализа
Анализ C#	2.2 млн	0:36:04
Анализ C# и Visual Basic .NET	3.7 млн	0:46:40
Прирост	68.1%	29.4%

Прирост времени анализа в процентном соотношении значительно ниже прироста объема анализируемого кода в процентном соотношении. Разница прироста может объясняться большей по сравнению с C# «многословностью» языка Visual Basic .NET.

Быстродействие анализа исходного кода на языке Visual Basic .NET было замерено для набора из 5 проектов с открытым исходным кодом суммарным объемом 1.6 млн. строк кода. Результаты тестирования приведены в табл. 4 вместе с объемами анализируемых проектов.

Кроме того, в таблице для проекта Roslyn дополнительно приведены замеры времени для анализа исходного кода на языках C# и Visual Basic .NET по отдельности.

Из таблицы видно, что время анализа не коррелирует с ростом объема кода. Это объясняется значительными различиями в анализируемых проектах. В табл. 5 приведено распределение времени по отдельным стадиям анализа для проектов Roslyn и EVE-IPH. Распределения для остальных проектов непоказательны, так как их анализ занимает малое время.

Табл. 4. Время анализа проектов.
Table 4. Projects analysis times.

Проект	**Кол-во строк**	**Кол-во предупр.**	**Время**
DeployOffice	968	26	0:00:12
JavaRa	2.5 тыс.	110	0:00:41
QuickVB	5.8 тыс.	99	0:00:19
EVE-IPH	86.8 тыс.	2759	0:17:39
Roslyn (Visual Basic .NET)	1.5 млн.	4907	0:20:53
Roslyn (C#)	2.2 млн.	7961	0:41:44

Табл. 5. Распределение времени по стадиям анализа.
Table 5. Distribution of time by stages of analysis.

Стадия анализа	**Roslyn**		**EVE-IPH**
	C#	**Visual Basic .NET**	
Синтаксический анализ и анализ графа вызовов	16.3%	15.9%	2.2%
Символьное исполнение и анализ потоков данных	80.1%	82.9%	36.8%
Анализ помеченных данных	3.6%	1.1%	61.0%

Можно заметить, что в случае проекта Roslyn распределения времени по стадиям анализа для обоих языков сходны, что объясняется единообразностью всего проекта вне зависимости от используемого языка. Однако в случае проекта EVE-IPH большую часть времени занимает стадия анализа помеченных данных. Это связано с тем, что этот проект имеет пользовательский интерфейс и большое количество полей для пользовательского ввода. Это порождает большое количество истоков для анализа помеченных данных и значительно увеличивает время анализа в целом.

Тестирование на проектах, не содержащих исходного кода на Visual Basic .NET, показало, что время анализа для них не изменилось, несмотря на значительные изменения в большинстве детекторов.

4.4 Качество автоматической классификации предупреждений

Для упрощения анализа и разметки предупреждений в SharpChecker предоставляется механизм автоматической классификации, основанный на методах машинного обучения. Для проверки работоспособности данного механизма использовались результаты, полученные при анализе набора из 5 проектов с открытым исходным кодом общим объемом более 1.6 млн. строк кода. Для оценки качества автоматической классификации были отобраны только размеченные предупреждения. Далее эти предупреждения были разделены на две выборки – обучающую и тестовую.

В процессе выполнения статического анализа был произведен сбор метрик для классов, методов, полей, свойств, а также различных локальных объявлений. Далее полученные метрики и обучающая выборка предупреждений были переданы механизму классификации для обучения. После обучения было выполнено предсказание на тестовой выборке предупреждений. Результаты предсказаний приведены в табл. 6.

Табл. 6. Результаты классификации для Visual Basic .NET.
Table 6. Classification results for Visual Basic .NET.

Результат классификации	**Истинные**	**Ложные**
Классифицированы как истинные	167	6
Классифицированы как ложные	24	40

Таким образом, доля верно классифицированных истинных предупреждений составила 87.4%, ложных 87.0%. Для языка C# доли верно предсказанных истинных и ложных предупреждений составляют соответственно 95.4% и 79.9%. Сравнение качества предсказаний для языков Visual Basic .NET и C# приведено в таблице 7.

Табл. 7. Сравнение результатов с предсказанием для C#.
Table 7. Comparison of results with prediction for C#.

Метрика	**Visual Basic .NET**	**C#**
Точность	87.3%	92.1%
Полнота	87.4%	95.4%
F_1-мера	87.4%	93.7%

Более низкое качество предсказания по сравнению с C# можно объяснить малым количеством размеченных предупреждений. Однако результаты позволяют сделать вывод, что механизм автоматической классификации применим к статическому анализу языка Visual Basic .NET.

5. Заключение

В рамках работы инструмент SharpChecker был доработан для поддержки статического анализа языка Visual Basic .NET. Разработанные методы адаптации детекторов позволили производить поиск большинства поддерживаемых для C# типов ошибок.

В отличие от прототипа, в данной работе была добавлена полная поддержка специфики языка Visual Basic .NET на всех этапах анализа и был реализован механизм сбора метрик исходного кода. Кроме того, было проведено масштабное тестирование качества статического анализа на наборе реальных проектов с открытым исходным кодом.

При проверке работы на реальных проектах суммарным объемом более 1.6 миллиона строк удалось обнаружить 7926 предупреждений. Доля истинных предупреждений превысила 84%, что сопоставимо с результатами для C#. Добавление поддержки языка Visual Basic .NET в промышленный статический анализатор SharpChecker не ухудшило его быстродействие на проектах, не содержащих исходного кода на языке Visual Basic .NET. Кроме того, были обнаружены межъязыковые предупреждения, связанные с исходным кодом на C# и на Visual Basic .NET. Также стоит отметить, что благодаря реализации механизма сбора метрик, стала возможна автоматическая разметка предупреждений для языка Visual Basic .NET.

Список литературы / References

[1]. Википедия. Visual Basic – Википедия, свободная энциклопедия, 2025. URL: https://ru.wikipedia.org/?curid=7394&oldid=143772950. [online; accessed 14.04.2025].

[2]. TIOBE Index for ranking the popularity of Programming languages, 2025. URL: https://www.tiobe.com/tiobe-index. [online; accessed 15.05.2025].

[3]. Википедия. Visual Basic .NET – Википедия, свободная энциклопедия, 2024. URL: https://ru.wikipedia.org/?curid=17042&oldid=140564863. [online; accessed 14.04.2025].

[4]. dotnet/roslyn: The Roslyn .NET compiler provides C# and Visual Basic languages with rich code analysis APIs. URL: https://github.com/dotnet/roslyn. [online; accessed 15.05.2025].

[5]. ГОСТ Р 56939-2024. Защита информации. Разработка безопасного программного обеспечения. Общие требования. 2024. URL: https://protect.gost.ru/document1.aspx?control=31&id=263523. [online; accessed 15.05.2025].

[6]. В. С. Карцев, В. Н. Игнатьев. Поддержка Visual Basic. NET в статическом анализаторе SharpChecker. Труды ИСП РАН, том 36, вып. 3, 2024 г., стр. 49–62. DOI: 10.15514/ISPRAS-2024-36(3)-4. / Karcev V.S., Ignatiev V.N. Support of Visual Basic .NET in SharpChecker Static Analyzer. Trudy ISP RAN/Proc. ISP RAS, 2024; vol. 36, issue 3, pp. 49-62 (in Russian). DOI: 10.15514/ISPRAS-2024-36(3)-4.

[7]. V. K. Koshelev, V. N. Ignatiev, A. I. Borzilov, A. A. Belevantsev. SharpChecker: Static analysis tool for C# programs. Programming and Computer Software, 43(4):268–276, 2017.

[8]. М. В. Беляев, Н. В. Шимчик, В. Н. Игнатьев, А. А. Белеванцев. Сравнительный анализ двух подходов к статическому анализу помеченных данных. Труды ИСП РАН, том 29, вып. 3, 2017 г., стр. 99–116. DOI: 10.15514/ISPRAS-2017-29(3)-7. / Belyaev M.V., Shimchik N.V., Ignatyev V.N., Belevantsev A.A. Comparative analysis of two approaches to the static taint analysis. Trudy ISP RAN/Proc. ISP RAS, 2017, vol. 29, issue 3, pp. 99-116 (in Russian). DOI: 10.15514/ISPRAS-2017-29(3)-7.

[9]. А. А. Белеванцев, Е. А. Велесевич. Анализ сущностей программ на языках Си/Си++ и связей между ними для понимания программ. Труды ИСП РАН, том 27, вып. 2, 2015 г., стр. 53–64. DOI: 10.15514/ISPRAS-2015-27(2)-4. / Belevantsev A., Velesevich E. Analyzing C/C++ code entities and relations for program understanding. Trudy ISP RAN/Proc. ISP RAS, 2015, vol. 27, issue 2, pp. 53–64. DOI: 10.15514/ISPRAS-2015-27(2)-4.

[10]. U. V. Tsiazhkorob, V. N. Ignatyev. Classification of Static Analyzer Warnings using Machine Learning Methods. 2024 Ivannikov Memorial Workshop (IVMEM):69–74, 2024. DOI: 10.1109/IVMEM63006.2024.10659704.

[11]. В. С. Карцев, В. Н. Игнатьев. Повышение точности статического анализа за счет учета значений полей класса, имеющих единственное константное значение. Труды ИСП РАН, том 34, вып. 6, 2022 г., стр. 29–40. DOI: 10.15514/ISPRAS-2022-34(6)-2. / Karcev V.S., Ignatiev V.N. Improving the accuracy of static analysis by accounting for the values of class fields that can have only one constant value. Trudy ISP RAN/Proc. ISP RAS, 2022, vol. 34, issue 6, pp. 29-40. DOI: 10.15514/ISPRAS-2022-34(6)-2.

[12]. Wikipedia contributors. List of tools for static code analysis – Wikipedia, The Free Encyclopedia, 2024. URL: https://en.wikipedia.org/w/index.php?title=List_of_tools_for_static_code_analysis&oldid=1218561224. [online; accessed 15.05.2025].

[13]. W. Wei, M. Yunxiu, H. Lilong, B. He. From source code analysis to static software testing. In 2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA), 1280–1283. IEEE, 2014.

[14]. E. Firouzi, A. Sami. Visual Studio Automated Refactoring Tool Should Improve Development Time, but ReSharper Led to More Solution-Build Failures. В 2019 IEEE Workshop on Mining and Analyzing Interaction Histories (MAINT), 2–6. IEEE, 2019.
[15]. ReSharper Features, 2025. URL: https://www.jetbrains.com/ru-ru/resharper/features/. [online; accessed 15.05.2025].
[16]. V. Lenarduzzi, F. Lomio, H. Huttunen, D. Taibi. Are SonarQube Rules Inducing Bugs? В 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), 501–511. IEEE, 2020.
[17]. G. A. Campbell, P. P. Papapetrou. SonarQube in action. Manning Publications Co., 2013.
[18]. VB.NET static code analysis | SonarQube, 2025. URL: https://rules.sonarsource.com/vbnet/. [online; accessed 15.05.2025].
[19]. Common Vulnerabilities | Kiuwan, 2025. URL: https://www.kiuwan.com/common-vulnerabilities/. [online; accessed 15.05.2025].
[20]. IOperation Interface (Microsoft.CodeAnalysis) | Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/api/microsoft.codeanalysis.ioperation?view=roslyn-dotnet-4.13.0. [online; accessed 19.05.2025].
[21]. R. Baldoni, E. Coppa, D. C. D'elia, C. Demetrescu, I. Finocchi. A Survey of Symbolic Execution Techniques. ACM Comput. Surv., 51(3), 2018. DOI: 10.1145/3182657.
[22]. Кошелев В.К., Игнатьев В.Н., Борзилов А.И. Инфраструктура статического анализа программ на языке C#. Труды ИСП РАН, том 28, вып. 1, 2016 г., стр. 21–40, DOI: 10.15514/ISPRAS-2016-28(1)-2. / Koshelev V., Ignatyev V., Borzilov A. C# static analysis framework. Trudy ISP RAN/Proc. ISP RAS, 2016, vol. 28, issue 1, pp. 21-40 (in Russian). DOI: 0.15514/ISPRAS-2016-28(1)-2.
[23]. EVE Isk per Hour. URL: https://eveiph.github.io/. [online; accessed 15.04.2025].

Информация об авторах / Information about authors

Вадим Сергеевич КАРЦЕВ – аспирант Физтех-школы Радиотехники и Компьютерных Технологий МФТИ, сотрудник ИСП РАН. Научные интересы: компиляторные технологии, статический анализ программ, статическое символьное выполнение, поиск дефектов в исходном коде.

Karcev Vadim KARCEV – postgraduate student of the Phystech School of Radio Engineering and Computer Technologies of MIPT, employee of the ISP RAS. Research interests: compiler technologies, static program analysis, static symbolic execution, searching for defects in source code.

Валерий Николаевич ИГНАТЬЕВ – кандидат физико-математических наук, старший научный сотрудник ИСП РАН, доцент кафедры системного программирования факультета ВМК МГУ. Научные интересы включают методы поиска ошибок в исходном коде ПО на основе статического анализа.

Valery Nikolaevich IGNATIEV – Cand. Sci. (Phys.-Math.), Senior Researcher at the ISP RAS, Associate Professor at the Department of System Programming at the Faculty of Computational Mathematics and Cybernetics at Moscow State University. Research interests include methods for finding errors in software source code based on static analysis.

DOI: 10.15514/ISPRAS-2025-37(6)-19

Hybrid Approach to Directed Fuzzing

[2] *D.A. Parygina, ORCID: 0000-0002-4029-0853 <parygina.darya.2001@yandex.ru>*
[1,2] *T.P. Mezhuev, ORCID: 0009-0009-0610-287X <mezhuevtp@ispras.ru>*
[1] *D.O. Kutz, ORCID: 0000-0002-0060-8062 <kutz@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*

[2] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia.*

Abstract. Program analysis and automated testing have recently become an essential part of SSDLC. Directed greybox fuzzing is one of the most popular automated testing methods that focuses on error detection in predefined code regions. However, it still lacks ability to overcome difficult program constraints. This problem can be well addressed by symbolic execution, but at the cost of lower performance. Thus, combining directed fuzzing and symbolic execution techniques can lead to more efficient error detection.
In this paper, we propose a hybrid approach to directed fuzzing with novel seed scheduling algorithm, based on target-related interestingness and coverage. The approach also performs minimization and sorting of objective seeds according to a target-related information. We implement our approach in Sydr-Fuzz tool using LibAFL-DiFuzz as directed fuzzer and Sydr as dynamic symbolic executor. We evaluate our approach with Time to Exposure metric and compare it with pure LibAFL-DiFuzz, AFLGo, and other directed fuzzers. According to the results, Sydr-Fuzz hybrid approach to directed fuzzing shows high performance and helps to improve directed fuzzing efficiency.

Keywords: directed greybox fuzzing; symbolic execution; hybrid fuzzing; program analysis.

For citation: Parygina D.A., Mezhuev T.P., Kutz D.O. Hybrid Approach to Directed Fuzzing. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 53-64. DOI: 10.15514/ISPRAS-2025-37(6)-19.

Acknowledgements. The results were obtained using the services of the Ivannikov Institute for System Programming (ISP RAS) Data Center.

Гибридный подход к направленному фаззингу

[2] *Д.А. Парыгина, ORCID: 0000-0002-4029-0853 <parygina.darya.2001@yandex.ru>*
[1,2] *Т.П. Межуев, ORCID: 0009-0009-0610-287X <mezhuevtp@ispras.ru>*
[1] *Д.О. Куц, ORCID: 0000-0002-0060-8062 <kutz@ispras.ru>*
[1] *Институт системного программирования им. В.П. Иванникова РАН, Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.*
[2] *Московский государственный университет имени М.В. Ломоносова, Россия, 119991, Москва, Ленинские горы, д. 1.*

Аннотация. Анализ программ и автоматизированное тестирование в последнее время стали неотъемлемой частью РБПО. Направленный фаззинг – один из самых популярных методов автоматизированного тестирования, который фокусируется на поиске ошибок в заранее определенных областях кода. Однако этот метод не способен преодолевать сложные программные ограничения. Эта проблема может быть эффективно решена с помощью символьного выполнения, но ценой более низкой производительности. Таким образом, комбинирование методов направленного фаззинга и символьного выполнения может привести к более эффективному поиску ошибок в программах.

В этой статье мы предлагаем гибридный подход к направленному фаззингу с оригинальным алгоритмом планирования входных данных, основанным на пользе для достижения целевых точек и увеличения покрытия кода. В подходе также выполняется минимизация и сортировка результатов анализа в соответствии с информацией о целевых точках. Мы реализовали наш подход в инструменте Sydr-Fuzz, используя LibAFL-DiFuzz в качестве направленного фаззера и Sydr в качестве динамического символьного исполнителя. Мы оценили наш подход с помощью метрики Time to Exposure и сравнили его с чистым LibAFL-DiFuzz, а также с инструментом AFLGo и другими направленными фаззерами. Согласно результатам, гибридный подход Sydr-Fuzz к направленному фаззингу демонстрирует высокую производительность и помогает повысить эффективность направленного фаззинга.

Ключевые слова: направленный фаззинг; символьное выполнение; гибридный фаззинг; анализ программ.

Для цитирования: Парыгина Д.А., Межуев Т.П., Куц Д.О. Гибридный подход к направленному фаззингу. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 53–64 (на английском языке). DOI: 10.15514/ISPRAS–2025–37(6)–19.

Благодарности: Результаты получены с использованием услуг Центра коллективного пользования Института системного программирования им. В.П. Иванникова РАН – ЦКП ИСП РАН.

1. Introduction

Program analysis and automated testing have recently become an essential part of SSDLC [1-3]. New software should be properly tested before releasing to reduce the probability of vulnerabilities.

Among various testing methods, fuzzing [4-5] still remains one of the most popular choices. It is a comparatively fast dynamic testing technique that allows checking program behavior on a wide range of input seeds. However, for certain specific goals, such as static analysis report verification, patch testing, crash reproduction, coverage-guided greybox fuzzing may appear less effective as its only metric to optimize is the global coverage value. Directed greybox fuzzing [6], in opposite, focuses on predefined program locations, called target points, to address these goals. In directed fuzzing, there are special proximity metrics that show how close a certain program execution is to reaching target points.

But for all the advantages of directed fuzzing approach, it still lacks the ability to overcome difficult program constraints, and thus can be unable to explore important code parts. Such problems are well addressed by symbolic execution [7-9] technique that interprets program instructions in terms of symbolic values associated with program variables, and discovers new program paths by inverting branch constraints. SMT-solvers [10-11] assistance allows symbolic executors solving complicated formulas and generate inputs that will lead the program along the desired execution path.

The main limitation of symbolic execution is its high overheads. Thus, both directed greybox fuzzing and symbolic execution can benefit from their combination. In this paper we propose hybrid directed fuzzing approach with novel seed exchange algorithm. We propose also post-analysis minimization and sorting of objectives (generated inputs considered to be analysis results) to determine which target points are reached, and what side effects (e.g., real program crashes or timeouts) are obtained.

This paper makes the following contributions:

- We propose a hybrid approach to directed fuzzing with novel seed scheduling algorithm, based on target-related interestingness and coverage. The approach also performs minimization and sorting of objective seeds according to a target-related information.
- We implement our approach in Sydr-Fuzz [12] tool using LibAFL-DiFuzz [13] as directed fuzzer and Sydr [9] as dynamic symbolic executor.
- We evaluate our approach with Time to Exposure metric and compare it with pure LibAFL-DiFuzz, AFLGo [6], BEACON [14], WAFLGo [15], WindRanger [16], FishFuzz [17], and Prospector [18]. We get 3 out of 7 best results with speedup up to 1.86 times from the second-best results. We get significant improvements on 3 out of 7 examples comparing with pure LibAFL-DiFuzz fuzzer with gaining speedup up to 4.9 times, and reaching an undiscovered target point.

2. Related Work

2.1 AFLGo

One of the first solutions in the field of directed greybox fuzzing was AFLGo [6] tool. The authors solve the problem of exploring given target points in code by scheduling energy of the inputs based on *simulated annealing* algorithm [19-20]. According to the algorithm, during energy scheduling, large energy values are assigned to inputs that provide the closest approximation to the target points.

AFLGo treats the target point exploration problem as an optimization problem. The resulting set of inputs with the largest energy values converges asymptotically to a set of global optimal solutions that allow to explore the program execution paths closest to the target points. To regulate the convergence rate, the authors use the exponential law of decreasing parameter *temperature*, which characterizes the probability of assigning large energy values to the worst solutions.

A metric based on the distances over the target basic blocks is proposed as a metric for the proximity of the program execution to reach the target points. Before fuzzing begins, a stage of static program instrumentation is performed: a program call graph and intra-procedural control flow graphs (CFGs) are constructed. Each vertex is matched with a metric value characterizing the distance from this vertex to all target vertices of the graph. The metric value for specific input is calculated as a normalized sum of metric values of each basic block in the program execution trace. The proposed metric allows to assign large energy values to those inputs that provide a shorter distance to the target points. However, such a solution has some drawbacks. First of all, due to the need for calculation metric values for all CFG basic blocks and all program functions, the stage of static instrumentation entails considerable time expenses and a large number of unnecessary calculations. At that, any changes in the target program require the static instrumentation stage to be performed anew. In addition, the solution converges to the selection of input data providing the minimum path length to the target positions without considering the execution context, indirect transitions, and program data flow.

2.2 BEACON

BEACON [14] uses *reachability analysis* to prune irrelevant basic blocks. It computes the reachability to the target point for every basic block and then the irrelevant ones are pruned, so they

are not being instrumented. Interesting blocks are used in further analysis to apply constraints on them. Such approach tends to reduce overheads significantly, however limits BEACON to support only single-target mode.

In the next phase the backward interval analysis is used. It slices the program using statically computed control flow information and then applies the analysis that builds value intervals for all variables in program basic blocks. These value intervals are used to construct formulas for target reachability depending on value constraints. Then it inserts such formulas in code as instrumentation. At runtime values are checked for these constraints, if so, then program proceeds further to target point, otherwise it terminates.

2.3 WAFLGo

WAFLGO [15] is a directed greybox fuzzer that was designed to discover bugs presented in new code commits. Traditional fuzzers usually focus only on reaching the commit-modified code, but they miss bugs that emerge in the wider affected code. WAFLGO solves this by introducing *Critical Code Guidance*, which separates *path-prefix code* (that leads to the change site) from *data-suffix code*, which represents the code influenced by the modified variables. It also has multi-target scenarios to balance precision and efficiency using a lightweight, function-level distance metric, effective input generation strategies using special mutation stages. By focusing the fuzzing effort on the critical code introduced in a commit, WAFLGo improves the efficiency and effectiveness of bug finding. It allows developers to quickly identify and fix vulnerabilities introduced by recent changes, reducing the risk of shipping buggy code.

2.4 WindRanger

WindRanger [16] is a directed greybox fuzzer that was designed to improve the efficiency of vulnerability discovery toward specific code regions (called *target sites*). Many greybox fuzzers treat all basic blocks the same, but WindRanger introduces the concept of *Deviation Basic Blocks* (*DBB*), the blocks that have the strongest impact on fuzzing planning, using them to calculate seed distances, guide mutations, and prioritize inputs more accurately. It also uses data flow analysis (*taint tracking*) to evaluate the difficulty of reaching conditions, enhancing both precision and efficiency. By actively identifying and mitigating path divergence, WindRanger improves the efficiency and effectiveness of DGF. It reduces the amount of time wasted exploring irrelevant code regions, allowing the fuzzer to focus on paths that are more likely to lead to the target and uncover vulnerabilities.

2.5 FishFuzz

FishFuzz [17] addresses the challenge of generating high-quality inputs for directed greybox fuzzing, specifically focusing on the creation and adaptation of a dictionary of "interesting" input tokens. It introduces an *adaptive dictionary creation* and update strategy. Instead of relying solely on seed inputs, it monitors the program's execution during fuzzing and extracts constant values used in comparisons (e.g., equality checks, greater-than/less-than). These values are then dynamically added to the dictionary. This allows FishFuzz to "fish" for relevant constants revealed during the fuzzing process itself. FishFuzz can optionally integrate with a constraint solver (Z3) to further refine and generalize the extracted constants, potentially discovering related values that can lead to new code coverage.

2.6 Prospector

Prospector [18] tackles the problem of imprecise call site targeting in directed greybox fuzzing. It introduces a novel approach that combines *precise call site identification* with *adaptive seed selection* for directed greybox fuzzing. Prospector uses a static analysis technique that is more

sensitive to the program's control flow and data flow context. This allows it to differentiate between different call sites of the target function by analyzing the code paths that lead to them. To further refine the call site identification, Prospector performs dynamic validation during fuzzing. It monitors the execution of the program and confirms whether the execution path actually leads to the intended call site based on input. Prospector uses the information from the static and dynamic analysis to prioritize seed inputs that are more likely to reach the correct call site. It ranks seed inputs based on their estimated distance to the target call site (considering the context information).

2.7 Hybrid directed fuzzing

In order to help directed fuzzer overcome complicated code constraints, several works propose a hybrid approach. The fuzzer is aided by symbolic or concolic executor that generates new inputs and discovers new code coverage.

In 1dVul [21], a dominator-based symbolic execution mechanism is involved when the fuzzer gets stuck for a specified time threshold. It takes the input with the shortest distance to the target point and tries to generate new ones that could reach target point dominators on the further paths.

DrillerGO [22] limits directive concolic execution to a subset of specific branch constraints. This subset is defined during backward pathfinding analysis that is based on searching suspicious API call strings in the recovered control flow graph.

Berry [23] and LeoFuzz [24] run concolic execution in parallel with the directed fuzzer sharing inputs via several input queues. The division is made based on inputs priority and capability of reaching target points, or discovering new coverage.

HyperGo [25] proposes an Optimized Symbolic Execution Complementary scheme for hybrid directed fuzzing. The scheme prunes the unreachable unsolvable branches, and prioritizes symbolic execution of the seeds whose paths are closer to the target and have more branches that are difficult to cover with the fuzzer.

Despite the variety of approaches, all of them are closed-source with no way to conduct comparison experiments.

3. Hybrid approach to directed fuzzing

In this paper, we present a novel hybrid approach to directed fuzzing. We combine directed greybox fuzzing and symbolic execution to share interesting inputs and benefit from each other. In this section, we describe the details of our integration approach.

3.1 Architecture overview

The overall integration scheme is shown in Fig. 1.

There are instances of directed greybox fuzzer and symbolic executor working in parallel and controlled by orchestrator instance. Orchestrator manages the whole analysis pipeline (described in section 4.1), validates configuration, launches fuzzer and symbolic executor instances, and provides their effective communication. After analysis, orchestrator sorts and minimizes found objectives – inputs that allow reaching specified target points.

Directed fuzzing tool LibAFL-DiFuzz [13] performs greybox fuzzing of the program focusing on reaching predefined target points. LibAFL-DiFuzz calculates proximity metric for each target program execution. Proximity metric is based on similarity between Enhanced Target Sequences (ETS) and the execution trace. ETS are built separately for each target point, and contain all the dominator functions and basic blocks of this target point.

Symbolic executor tool Sydr [9] performs concolic execution of the program. During the concrete execution of the tested program on some input, Sydr collects symbolic path constraints. They are used to invert branches by building corresponding path predicates and validating them with SMT-solver [11]. For all successful solutions new inputs are generated.

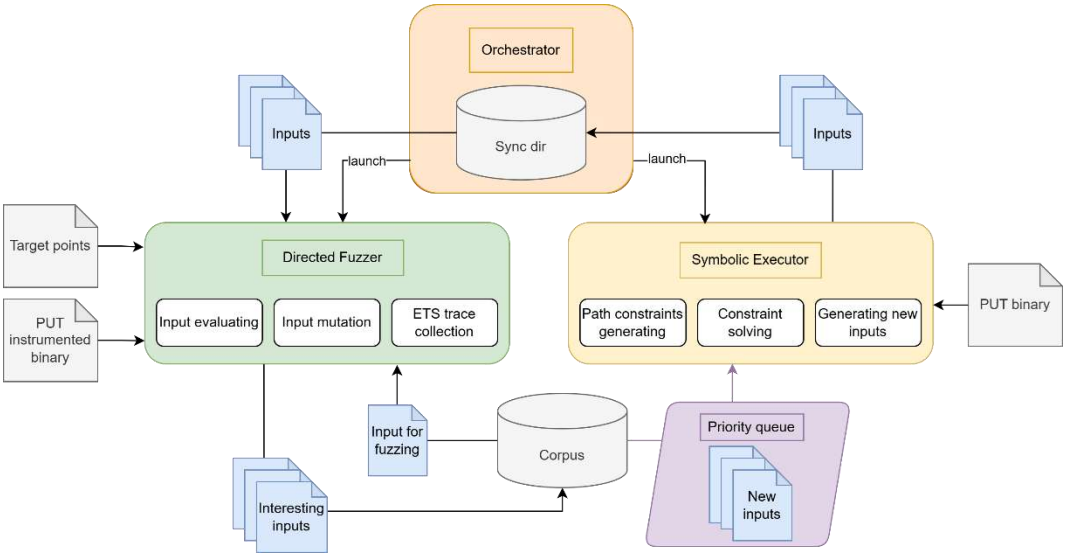


Fig. 1. Hybrid directed fuzzing scheme.

3.2 Seed scheduling algorithm

In hybrid fuzzing, fuzzer and symbolic executor are running in parallel and sharing input seeds. Symbolic executor puts generated inputs in a separate directory called *sync dir*. LibAFL-DiFuzz performs synchronization with symbolic executor at dynamically calculated time intervals by analyzing all seeds from this directory. It measures whether each seed is interesting for ETS metric or explores new coverage, and if either is true, saves it into the fuzzing corpus directory *corpus*. In case an input causes the program to crash or hang, or leads it to one of the specified target points, it is saved into an *objective directory*. Additionally, LibAFL-DiFuzz has a feature to import all seeds from symbolic executor without preliminarily measuring them. In some cases, the fuzzer cannot produce many inputs on specific target programs, thus adding all seeds can give a boost to find new coverage paths or improve ETS metric.

Scheduling inputs for symbolic execution is another part of seed exchanging algorithm. Since Sydr analyzes only one program execution path at a time, the quality of selected seed is crucial for hybrid fuzzing performance. All seeds generated by the fuzzer and stored in its *corpus* directory are ordered by the orchestrator with the help of *priority queue* (see section 4.2). The seeds are ranked by *interest* score, which is calculated based on ETS exploration and coverage growth. The input is assigned higher priority if it is useful for both ETS and coverage exploration. After that, the seed is taken from the queue and analyzed by symbolic executor. In case of successful path inversion, the modified input is put into the *sync dir*.

Synchronization interval for the fuzzer is determined dynamically to improve fuzzing performance. Default time is 60 seconds, but after the first synchronization it is computed as *max(60, **t***3)*, where ***t*** is the time in seconds that was spent on importing all seeds from symbolic executor. This heuristic is needed to compensate a possibly long synchronization and allocate more processor time on fuzzing process.

3.3 Objective sorting and minimization

In directed fuzzing, an input that is considered objective can be one of three types: 1) input that reaches one or more target points without crashing the program; 2) a real crash; 3) a timeout (set to 10 seconds by default). For each objective its type is explicitly specified in the corresponding

metadata file created by LibAFL-DiFuzz (see section 4.3). The list of reached target points is also saved in this metadata file.

Once the fuzzing is complete, all found objectives can be minimized and sorted by analyzing their metadata files. Objective minimization is based on ETS novelty. During the target program execution on certain input data, the fuzzer collects an ETS trace – a sequence of IDs of visited basic blocks belonging to at least one ETS. This trace is also saved to the corresponding metadata file. If several objectives have the same ETS traces, we can keep only one of them since only unique ETS traces can bring new useful feedback. Thus, all objectives can be clustered by ETS traces, and only one from each cluster can be taken. This will result in a minimized objective set with variety of ETS traces.

After minimization part is done, remaining objectives are sorted by reached target points. We create a separate directory for every reached target point and copy all corresponding objectives there.

4. Implementation

We implemented our hybrid approach to directed fuzzing with Sydr-Fuzz [12] tool as orchestrator, LibAFL-DiFuzz [13] as directed fuzzer, and Sydr [9] as dynamic symbolic executor. In this section, we describe practical details of the integration.

4.1 Sydr-Fuzz pipeline

Sydr-Fuzz tool allows to run hybrid fuzzing by launching such fuzzers as libFuzzer [1], AFL++ [2], or Honggfuzz [26] alongside with Sydr symbolic executor. For directed fuzzing we implemented a hybrid fuzzing approach that allows Sydr-Fuzz combining LibAFL-DiFuzz and Sydr.

Analysis specification is set by configuration TOML-files that match the following sample:

```
[sydr]
args = "-j 4"
target = "/target_sydr @@"
jobs = 3

[difuzz]
target = "/target_libafl_difuzz @@"
args = "-j2 -i /corpus -e /ets.toml"
path = "/fuzzer_libafl_difuzz"
```

This sample Sydr-Fuzz configuration specifies two tables – **[sydr]** and **[difuzz]**, containing parameters for launching each tool, respectively. Every table specifies arguments to run a tool, and a binary program to analyze. Symbolic executor Sydr requires the program to be built without any instrumentation (except debugging information *-g*). For directed fuzzer LibAFL-DiFuzz the program should be compiled with specific fuzzer instrumentation that allows tracking ETS traces. Directed fuzzer also needs its own configuration *config.toml* that specifies target points. To improve analysis performance, Sydr-Fuzz can also launch several instances of each tool in parallel (*jobs* parameter).

Once Sydr-Fuzz has been started with the specified configuration, it launches the fuzzer and verifies that all parallel clients specified by *jobs* parameter have been successfully started. It also enables *watchdog* to monitor a fuzzing *corpus* and track new files. After that, the actual fuzzing stage begins. There are four main tasks performed by Sydr-Fuzz during the hybrid directed fuzzing with LibAFL-DiFuzz and Sydr:

1. Sydr-Fuzz constantly keeps launching the specified number of Sydr workers. For each worker the most interesting seed is pulled from the *priority queue* (see section 4.2). All inputs generated by Sydr are stored into *sync dir* that is analyzed by LibAFL-DiFuzz directly.

2. Every second Sydr-Fuzz parses a fuzzer log and prints a basic information from every client: the number of files in its corpus, the number of objectives found, execution speed, etc.
3. Every minute Sydr-Fuzz updates the *priority queue* with new files from fuzzing *corpus*, which are received from *watchdog*. It also checks conditions to stop analysis, such as time without new coverage, or reaching all target points. Sydr-Fuzz prints analysis statistics (total number of files in the corpus, total number of objectives, average execution speed, last saved objective/corpus time, etc.), and the list of unique reached target points.
4. When Sydr-Fuzz receives a signal to terminate, all LibAFL-DiFuzz clients and Sydr-workers are killed, the final statistics are printed. After that, all files from the *objective directory* are minimized and sorted according to the algorithm described in section 3.3.

4.2 Seed priority queue

The *priority queue* is organized as a binary heap, where the seeds are stored along with a special data structure describing input priority. This structure has three fields:

1) *is_interestring_ets* flag shows whether the input is interesting for ETS;
2) *is_interesting_map* flag shows whether it is interesting for program coverage;
3) *file score* metric based on file system metadata: file creation time / file size. Let us call the first two flags *ETS score* and *coverage score*, respectively.

At runtime, Sydr-Fuzz collects a list of newly found inputs from the *corpus* once a minute. For each of them, *ETS* and *coverage scores* are obtained from LibAFL-DiFuzz metadata described in section 4.3. The *file score* is calculated based on file system metadata, and a final structure with three fields is constructed. Since the *priority queue* is a binary heap, when a new seed is added, it is placed in the correct sorted order. That is, when scheduling inputs for symbolic execution, the most priority is given to seeds that discover new ETS traces (interesting for directed fuzzing). The next priority is assigned to seeds that discover new program coverage. A *file score* is a metric that allows choosing new file according to its size. While it is important to always choose the newest file in order to produce the most relevant inputs for the fuzzer, symbolic executor may be less effective on large inputs.

4.3 LibAFL-DiFuzz seed metadata

For the effective communication between LibAFL-DiFuzz and Sydr, all important input information is saved in a separate metadata file next to the input. This file includes fields used for calculating seed priority when adding new seed to Sydr *priority queue*, and also fields used for crash sorting and minimization.

Seed priority is based on *ETS score* and *coverage score* (see section 4.2) that are saved by LibAFL-DiFuzz in the form of flags. Whether the seed is interesting for ETS or program coverage, is determined when evaluating LibAFL modules ETSFeedback and MapFeedback after target program execution, respectively.

Objective minimization is based on ETS trace values that are also saved in metadata files. ETS trace is kept as a list containing IDs of visited basic blocks belonging to at least one ETS. And for sorting, we need to know which target points are reached during execution on each objective file. The list of reached target points is saved to metadata file too, and each list element keeps source file name and line number of the target point. For the further analysis and results presentation, we need to know whether the objective is a real crash, or timeout, or just the one that leads to reaching some target points. For these reasons, two flags are saved in metadata: *is_crash* and *is_timeout*. They are set when evaluating LibAFL modules CrashFeedback and TimeoutFeedback, respectively. All seeds in *objective directory* with both flags set to false are considered to be just reaching target points.

5. Evaluation

We evaluated our hybrid approach using TTE (Time to Exposure) metric that shows the time spent on crash discovering. Experimental setup included one machine with two 32-core AMD EPYC 7542 CPUs, 512 GB of RAM, and Ubuntu 20.04 LTS. We compared our hybrid directed fuzzer (Sydr-Fuzz) with pure LibAFL-DiFuzz and other directed fuzzers, such as AFLGo, BEACON, WindRanger, WAFLGo, FishFuzz, and Prospector. For the evaluation, we selected five different projects from the AFLGo experimental set. These projects have publicly known CVEs in specific versions with TTE no greater than 150 minutes on AFLGo. The set of chosen target points with their IDs, locations, and experiment timeouts, is shown in Table 1.

Table 1. Target points chosen for experiments.

Project	Target point ID	Location	TimeOut
cxxfilt	$cxxfilt_1$	libiberty/cp-demangle.c: 1596	45 min
	$cxxfilt_2$	libiberty/cplus-dem.c: 4319	
giflib	giflib	util/gifsponge.c: 61	30 min
jasper	jasper	src/libjasper/mif/mif_cod.c:491	30 min
libming	libming	util/outputscript.c:1493	150 min
libxml	$libxml_1$	valid.c:1181	60 min
	$libxml_2$	valid.c:1189	

Each experiment was run several times with a defined timeout. In Sydr-Fuzz experiments, one instance each of LibAFL-DiFuzz and Sydr were launched simultaneously and worked in parallel. Sydr inverted branches in direct order with one solving thread. Every Sydr process was limited up to 2 minutes, with a 10-second timeout for solving a single SMT-query and a 60-second total solving time. Strong optimistic solutions [27] and symbolic address fuzzing [28] were enabled for Sydr. For each execution, we measured time intervals between the experiment start and different crashes discovering. After the end of experiment, we checked crash seeds with *gdb* to validate they were real crashes. Experiments results for the best attempts with outliers filtered are shown in Table 2. Columns represent different instruments, lines – different target points, and each cell contains TTE value in seconds. Empty cells mean that the instrument was not able to reach target point.

Table 2.TTE experiments results.

TP ID	Sydr-Fuzz, s	LibAFL-DiFuzz, s	AFLGo, s	BEACON, s	WAFLGo, s	WindRanger, s	FishFuzz, s	Prospector, s
$cxxfilt_1$	**22**	175	49	60	41	44	384	470
$cxxfilt_2$	136	-	63	**55**	3043	704	1208	347
giflib	**3**	**3**	18	4	5	55	41	45
jasper	18	**16**	**16**	50	-	81	287	22
libming	119	127	412	**53**	-	-	862	59
$libxml_1$	**8**	14	28	43	107	666	2285	1694
$libxml_2$	61	300	343	**44**	-	1108	-	-

TTE results show that Sydr-Fuzz hybrid directed fuzzer outperforms all other directed fuzzers on 3 out of 7 examples: **cxxfilt$_1$**, **giflib**, and **libxml$_1$** with speedup up to 1.86 times. For **jasper** and **libxml$_2$** Sydr-Fuzz results are second after the best measurements, and the difference is relatively small. As we see in Table 2, BEACON is another tool that has 3 out of 7 best results, however it supports only single-target mode, in opposite to all other tools. This can have direct influence on better TTE results, since in multi-target projects (**cxxfilt**, **libxml**) other tools spend their energy on reaching several target points simultaneously.

Comparing Sydr-Fuzz approach with pure LibAFL-DiFuzz, we can see improvements for all examples except **jasper**. This is caused by CPU load introduced by Sydr-Fuzz extra actions (Sydr launching, synchronization) since TTE for **jasper** is relatively short. Since the first synchronization with Sydr is performed after 60 seconds, improvements for **cxxfilt$_1$**, **giflib**, and **libxml$_1$** can be mainly explained by the randomness of fuzzing process. However, for **cxxfilt$_2$**, **libming**, and **libxml$_2$** hybrid approach shows a real advantage with speedup up to 4.9 times. What is more, Sydr-Fuzz has managed to reach target point **cxxfilt$_2$** in a relatively short time, whereas pure LibAFL-DiFuzz has failed to reach it at all within 45 minutes timeout.

6. Conclusion

In this paper, we propose a hybrid approach to directed fuzzing with novel seed scheduling algorithm, based on target-related interestingness and coverage. The approach also performs minimization and sorting of objective seeds according to the target-related information. We implement our approach in Sydr-Fuzz tool using LibAFL-DiFuzz as directed fuzzer and Sydr as dynamic symbolic executor.

We evaluate our approach with Time to Exposure metric and compare it with pure LibAFL-DiFuzz, AFLGo, BEACON, WAFLGo, WindRanger, FishFuzz, and Prospector. We get 3 out of 7 best results with speedup up to 1.86 times from the second-best results. We get significant improvements on 3 out of 7 examples comparing with pure LibAFL-DiFuzz fuzzer with speedup up to 4.9 times, and reaching an undiscovered target point. To sum up, Sydr-Fuzz hybrid approach to directed fuzzing shows high performance and helps to improve directed fuzzing efficiency.

References

[1]. Howard M., Lipner S. The security development lifecycle. Microsoft Press Redmond (online), vol. 8, 2006. Available at: http://msdn.microsoft.com/en-us/library/ms995349.aspx, accessed 22.05.2025.

[2]. ISO/IEC 15408-3:2008: Information technology – Security techniques – Evaluation criteria for IT security – Part 3: Security assurance components. ISO Geneva (online), 2008. Available at: https://www.iso.org/standard/46413.html, accessed 22.05.2025.

[3]. GOST R 56939-2016: Information protection. Secure software development. General requirements. National Standard of Russian Federation (online), 2016. Available at: http://protect.gost.ru/document.aspx?control=7&id=203548, accessed 22.05.2025.

[4]. Serebryany K. Continuous fuzzing with libFuzzer and AddressSanitizer. 2016 IEEE Cybersecurity Development (SecDev), 2016, p. 157. DOI: 10.1109/secdev.2016.043.

[5]. Fioraldi A., Maier D., Eißfeldt H., Heuse M. AFL++: Combining incremental steps of fuzzing research. 14th USENIX Workshop on Offensive Technologies (WOOT 20), 2020.

[6]. Bohme M., Pham V.-T., Nguyen M.-D., Roychoudhury A. Directed greybox fuzzing. Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, 2017, pp. 2329–2344.

[7]. Shoshitaishvili Y., Wang R., Salls C., Stephens N., Polino M., Dutcher A., Grosen J., Feng S., Hauser C., Kruegel C., Vigna G. SOK: (state of) the art of war: Offensive techniques in binary analysis. 2016 IEEE Symposium on Security and Privacy (SP), 2016, pp. 138–157. DOI: 10.1109/SP.2016.17.

[8]. Borzacchiello L., Coppa E., Demetrescu C. FUZZOLIC: Mixing fuzzing and concolic execution. Computers & Security, 2021, vol. 108, p. 102368.

[9]. Vishnyakov A., Fedotov A., Kuts D., Novikov A., Parygina D., Kobrin E., Logunova V., Belecky P., Kurmangaleev S. Sydr: Cutting edge dynamic symbolic execution. 2020 Ivannikov ISPRAS Open Conference (ISPRAS), 2020, pp. 46–54. DOI: 10.1109/ISPRAS51486.2020.00014.

[10]. De Moura L., Bjørner N. Z3: An efficient SMT solver. Tools and Algorithms for the Construction and Analysis of Systems, 2008, pp. 337–340. DOI: 10.1007/978-3-540-78800-3_24.
[11]. Niemetz A., Preiner M. Bitwuzla at the SMT-COMP 2020. CoRR, vol. abs/2006.01621, 2020.
[12]. Vishnyakov A., Kuts D., Logunova V., Parygina D., Kobrin E., Savidov G., Fedotov A. Sydr-Fuzz: Continuous hybrid fuzzing and dynamic analysis for security development lifecycle. 2022 Ivannikov ISPRAS Open Conference (ISPRAS), 2022, pp. 111-123. DOI: 10.1109/ISPRAS57371.2022.10076861.
[13]. Parygina D., Mezhuev T., Kuts D. LibAFL-DiFuzz: Advanced Architecture Enabling Directed Fuzzing. 2024 Ivannikov ISPRAS Open Conference (ISPRAS), 2024. DOI: 10.1109/ISPRAS64596.2024.10899166.
[14]. Huang H., Guo Y., Shi Q., Yao P., Wu R., Zhang C. Beacon: Directed grey-box fuzzing with provable path pruning. 2022 IEEE Symposium on Security and Privacy (SP), 2022, pp. 36–50.
[15]. Xiang Y., Zhang X., Liu P., Ji S., Liang H., Xu J., Wang W. Critical code guided directed greybox fuzzing for commits. 33rd USENIX Security Symposium (USENIX Security 24), 2024, pp. 2459-2474.
[16]. Du Z., Li Y., Liu Y., Mao B. Windranger: A directed greybox fuzzer driven by deviation basic blocks. Proceedings of the 44th International Conference on Software Engineering, 2022, pp. 2440-2451.
[17]. Zheng H., Zhang J., Huang Y., Ren Z., Wang H., Cao C., Zhang Y., Toffalini F., Payer M. FISHFUZZ: Catch deeper bugs by throwing larger nets. 32nd USENIX Security Symposium (USENIX Security 23), 2023, pp. 1343-1360.
[18]. Zhang Z., Chen L., Wei H., Shi G., Meng D. Prospector: Boosting Directed Greybox Fuzzing for Large-Scale Target Sets with Iterative Prioritization. Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis, 2024, pp. 1351-1363.
[19]. Kirkpatrick S., Gelatt Jr C. D., Vecchi M. P. Optimization by simulated annealing. Science, 1983, vol. 220, no. 4598, pp. 671–680.
[20]. Cerny V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of optimization theory and applications, 1985, vol. 45, pp. 41–51.
[21]. Peng J., Li F., Liu B., Xu L., Liu B., Chen K., Huo W. 1dvul: Discovering 1-day vulnerabilities through binary patches. 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2019, pp. 605-616.
[22]. Kim J., Yun J. Poster: Directed hybrid fuzzing on binary code. Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 2637-2639.
[23]. Liang H., Jiang L., Ai L., Wei J. Sequence directed hybrid fuzzing. 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering, 2020, pp. 127-137.
[24]. Liang H., Yu X., Cheng X., Liu J., Li J. Multiple targets directed greybox fuzzing. IEEE Transactions on Dependable and Secure Computing, 21(1), 2023, pp. 325-339.
[25]. Lin P., Wang P., Zhou X., Xie W., Lu K., Zhang G. HyperGo: Probability-based directed hybrid fuzzing. Computers & Security, 142, 2024, p. 103851.
[26]. Swiecki R., Grobert F. Honggfuzz. Available at: https://github.com/google/honggfuzz, accessed 22.05.2025.
[27]. Parygina D., Vishnyakov A., Fedotov A. Strong optimistic solving for dynamic symbolic execution. 2022 Ivannikov Memorial Workshop (IVMEM), 2022, pp. 43-53. DOI: 10.1109/IVMEM57067.2022.9983965.
[28]. Kuts D. Towards symbolic pointers reasoning in dynamic symbolic execution. 2021 Ivannikov Memorial Workshop (IVMEM), 2021, pp. 42-49. DOI: 10.1109/IVMEM53963.2021.00014.

Информация об авторах / Information about authors

Дарья Алексеевна ПАРЫГИНА – магистр Московского государственного университета имени М.В. Ломоносова. Сфера научных интересов: символьное выполнение, гибридный фаззинг, направленный фаззинг.

Darya Alekseevna PARYGINA – master of Lomonosov Moscow State University. Research interests: symbolic execution, hybrid fuzzing, directed fuzzing.

Тимофей Павлович МЕЖУЕВ – бакалавр Московского государственного университета имени М.В. Ломоносова, лаборант Института системного программирования. Сфера научных интересов: символьное выполнение, гибридный фаззинг, направленный фаззинг.

Timofey Pavlovich MEZHUEV – bachelor of Lomonosov Moscow State University, laborant of the Institute for System Programming of the RAS. Research interests: symbolic execution, hybrid fuzzing, directed fuzzing.

Даниил Олегович КУЦ – кандитат технических наук, младший научный сотрудник Института системного программирования. Сфера научных интересов: динамический анализ, фаззинг, символьное выполнение, гибридный фаззинг.

Daniil Olegovich KUTZ – Cand. Sci. (Tech.), junior research assistant of the Institute for System Programming of the RAS. Research interests: dynamic analysis, fuzzing, symbolic execution, hybrid fuzzing.

DOI: 10.15514/ISPRAS-2025-37(6)-20

Итеративное обучение со слабым контролем с уточнением функций разметки на основе больших языковых моделей

[1,2] *А.Д. Сосновиков, ORCID: 0009-0004-1447-532X <sosnovikov.artur@gmail.com>*
[2] *А.Д. Земеров, ORCID: 0009-0006-4832-7610 <zemerov@tochka.com>*
[1] *Д.Ю. Турдаков, ORCID: 0000-0001-8745-0984 <turdakov@ispras.ru>*

[1] *Институт системного программирования РАН,*
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

[2] *Банк Точка,*
Россия, 109044, г. Москва, 3-й Крутицкий пер., д. 11.

Аннотация. Обучение высококачественных классификаторов в условиях ограниченного количества размеченных данных является одной из фундаментальных проблем машинного обучения. Несмотря на то, что большие языковые модели (LLM) демонстрируют впечатляющие результаты при решении задач классификации явного обучения (zero-shot), их прямое применение на практике затруднено из-за высокой вычислительной стоимости, чувствительности к формулировкам запросов (prompt engineering) и ограниченной интерпретируемости. В качестве масштабируемой альтернативы выступает обучение со слабым контролем, которое основано на объединении множества неточных функций разметки (labeling functions, LF). Однако создание и последующая настройка таких функций обычно требует существенных затрат ручного труда. В данной работе мы предлагаем подход LLM-Guided Iterative Weak Labeling (LGIWL), который сочетает генерацию функций разметки с помощью больших языковых моделей и методику обучения со слабым контролем в рамках итеративного цикла обратной связи. Вместо прямого использования LLM в качестве классификатора, мы применяем её для автоматического создания и постепенного уточнения функций разметки на основе ошибок промежуточного классификатора. Полученные функции фильтруются с использованием небольшого размеченного набора данных и затем применяются к неразмеченной выборке при помощи генеративной модели меток. Это позволяет обучить итоговый дискриминативный классификатор высокого качества при минимальных затратах на ручную аннотацию. Эффективность предложенного подхода продемонстрирована на реальной задаче классификации диалогов службы поддержки клиентов на русском языке. LGIWL существенно превосходит как классические эвристики на основе ключевых слов (Snorkel), так и подходы zero-shot на основе GPT-4, а также полностью контролируемый классификатор CatBoost, обученный на размеченных данных аналогичного размера. В частности, вариант LGIWL с моделью RuModernBERT достигает высокого показателя полноты при значительном улучшении точности, демонстрируя итоговый результат по метрике F1 = 0.863. Полученные результаты подтверждают как высокую устойчивость метода, так и его практическую применимость в условиях ограниченных ресурсов размеченных данных.

Ключевые слова: обучение со слабым контролем; финансовый сектор; модели LLM.

Для цитирования: Сосновиков А.Д., Земеров А.Д., Турдаков Д.Ю. Итеративное обучение со слабым контролем с уточнением функций разметки на основе больших языковых моделей. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 65–76. DOI: 10.15514/ISPRAS–2025–37(6)–20.

Iterative Weak Supervision with LLM-Guided Labeling Function Refinement

[1,2] *A.D. Sosnovikov, ORCID: 0009-0004-1447-532X <sosnovikov.artur@gmail.com>*
[2] A.D. Zemerov, *ORCID: 0009-0006-4832-7610 <zemerov@tochka.com>*
[1] *D.Y. Turdakov, ORCID: 0000-0001-8745-0984 <turdakov@ispras.ru>*

[1] *Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*

[2] *Tochka Bank, 11, 3-y Krutitskiy pereulok, Moscow, 109044, Russia.*

Abstract. Training high-quality classifiers in domains with limited labeled data remains a fundamental challenge in machine learning. While large language models (LLMs) have demonstrated strong zero-shot capabilities, their use as direct predictors suffers from high inference cost, prompt sensitivity, and limited interpretability. Weak supervision, in contrast, provides a scalable alternative through the aggregation of noisy labeling functions (LFs), but authoring and refining these rules traditionally requires significant manual effort. We introduce LLM-Guided Iterative Weak Labeling (LGIWL), a novel framework that integrates prompting with weak supervision in an iterative feedback loop. Rather than using an LLM for classification, we use it to synthesize and refine labeling functions based on downstream classifier errors. The generated rules are filtered using a small development set and applied to unlabeled data via a generative label model, enabling high-quality training of discriminative classifiers with minimal human annotation. We evaluate LGIWL on a real-world text classification task involving Russian-language customer service dialogues. Our method significantly outperforms keyword-based Snorkel heuristics, zero-shot prompting with GPT-4, and even a supervised CatBoost classifier trained on a full labeled dev set. In particular, LGIWL achieves strong recall while yielding a notable improvement in precision, resulting in a final F1 score of 0.863 with a RuModernBERT classifier–demonstrating both robustness and practical scalability.

Keywords: weak supervison; financial sector; LLM.

For citation: Sosnovikov A.D., Zemerov A.D., Turdakov D.Yu. Automating the process of responding to a tax request using weak supervision. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025. pp. 65-76 (in Russian). DOI: 10.15514/ISPRAS-2025-37(6)-20.

1. Введение

Обучение с учителем стало доминирующим подходом к обучению высокоэффективных классификаторов в задачах обработки естественного языка. Однако его успешность критически зависит от наличия большого объёма качественно размеченных данных – условия, которое редко выполняется на практике, особенно в специализированных областях, таких как финансы, здравоохранение и клиентская поддержка. Ручная аннотация является дорогостоящей, медленной и зачастую требует значительной предметной экспертизы.

Обучение со слабым контролем предлагает убедительную альтернативу, позволяя специалистам программно задавать эвристики разметки – так называемые функции разметки (labeling functions, LF), которые могут быть применены к большим неразмеченным корпусам. Эти слабые сигналы затем агрегируются с помощью вероятностных моделей для получения шумных, но масштабируемых обучающих меток. Фреймворки, такие как Snorkel, показывают, что при аккуратном проектировании функций разметки модели, обученные на слабо размеченных данных, могут приближаться по качеству к моделям, обученным на полностью ручной разметке. Однако разработка функций разметки остаётся нетривиальной задачей: она требует времени, экспертизы и итеративной доработки, что ограничивает доступность и масштабируемость обучения со слабым контролем на практике.

Тем временем, большие языковые модели (LLM), такие как GPT-4, продемонстрировали выдающиеся способности, позволяя решать новые задачи при минимальном (few-shot) или

полном отсутствии (zero-shot) размеченных данных. Однако использование LLM напрямую в качестве классификаторов порождает новые сложности: вывод (inference) требует значительных вычислительных ресурсов, предсказания чувствительны к формулировке подсказок, а отсутствие интерпретируемости мешает их интеграции в регламентированные или критически важные для производства системы.

В данной работе мы предлагаем новый подход, объединяющий семантическую гибкость больших языковых моделей LLM и надёжность с интерпретируемостью обучения со слабым контролем. Мы представляем метод *LLM-Guided Iterative Weak Labeling* (LGIWL) – фреймворк, в котором модели LLM рассматриваются не как чёрные ящики, а как генераторы функций разметки. LGIWL работает в итеративном цикле: ошибки классификатора используются для формирования подсказок LLM с целью генерации новых или уточнённых функций разметки; полученные функции фильтруются на небольшом валидационном наборе, и их выходы агрегируются для повторного обучения классификатора. Этот процесс, основанный на обратной связи, позволяет автоматически расширять и улучшать множество функций разметки, снижая необходимость в ручной настройке подсказок и разработке правил.

Мы применяем метод LGIWL к реальной задаче классификации обращений в службу поддержки в финансовой жносфере и демонстрируем, что он значительно превосходит метод запросов без явного обучения (zero-shot prompting), статическое обучение со слабым контролем и даже полностью супервизированные модели, обученные на таком же объёме размеченных данных. В частности, LGIWL достигает высокой полноты при существенном росте точности, демонстрируя свою эффективность как масштабируемого и интерпретируемого подхода к обучению в условиях ограниченных ресурсов.

2. Обзор релевантных работ

Программируемое обучение со слабым контролем. Обучение со слабым контролем позволяет создавать размеченные наборы данных, заменяя ручную аннотацию эвристиками или правилами, известными как функции разметки LF. Фреймворк Snorkel ввёл формализованный подход к этому процессу, моделируя точность и корреляцию между функциями разметки с помощью генеративной модели меток [1]. Это позволило агрегировать шумные и потенциально конфликтующие слабые метки. Последующие работы расширили Snorkel для применения в крупномасштабных задачах [2] и разработали методы для выявления структуры между функциями разметки [3]. Позже набор WRENCH стандартизировал протоколы оценки для слабого обучения [4].

Однако, несмотря на существенную автоматизацию моделирования меток, написание функций разметки остаётся ручной и трудоёмкой задачей, часто требующей предметной экспертизы и итеративной доработки. Системы, такие как Snuba [5] и WITAN [6], стремились уменьшить нагрузку на разработку LF путём генерации правил из начальных примеров или обратной связи пользователя. Тем не менее, многие из этих систем по-прежнему требуют участия человека в цикле или зависят от жёстко заданной логики правил.

Разметка на основе подсказок с использованием LLM. Большие языковые модели (LLM) предлагают новый путь для генерации правил путём интерпретации подсказок на естественном языке. В работе [7] было предложено использовать LLM для создания слабых меток путём постановки структурированных вопросов, рассматривая выходные данные LLM как источник шумной супервизии. Система Alfred [8] развила эту идею, применяя подсказки на естественном языке в качестве интерфейса для генерации и управления функциями разметки. Другие системы, такие как ScriptoriumWS [9], ещё больше автоматизируют процесс, преобразуя подсказки в исполняемые функции разметки на языке Python.

Недавно появились полностью итеративные фреймворки. Например, DataSculpt [10] использует LLM для предложения новых функций разметки на основе few-shot подсказок и

уточняет набор правил в ходе нескольких раундов обучения и оценки. PRBoost [11] предложил цикл в стиле бустинга, где сгенерированные правила оцениваются и постепенно добавляются для повышения точности классификации.

Вклад данной работы. Наш метод LGIWL опирается на эти разработки, но вводит ключевое новшество: итеративное уточнение функций разметки, сгенерированных LLM, на основе сигналов об ошибках классификатора, встроенное в структурированную архитектуру обучения со слабым контролем. В отличие от предыдущих методов, которые рассматривают генерацию подсказок или правил как одноразовые шаги, LGIWL замыкает цикл между подсказками LLM, агрегацией меток и обучением дискриминативной модели. Это обеспечивает масштабируемую и адаптивную слабую разметку в реалистичных условиях с ограниченными ресурсами и минимальным участием человека.

3. Постановка задачи

Рассмотрим стандартную задачу бинарной классификации, в которой требуется построить функцию $f_\theta : X \to \{0,1\}$, предсказывающую принадлежность текстового объекта $x \in X$ к заданной семантической категории (например, обращению, связанному с тарифом). При типичном подходе с учителем для обучения модели необходим полностью размеченный набор данных $\mathcal{D}_{\text{sup}} = \{(x_i, y_i)\}^N_{i=1}$. Однако на практике получение такого набора часто затруднено из-за высокой стоимости ручной аннотации, нехватки квалифицированных специалистов или малого количества примеров целевого класса.

Вместо этого будем предполагать, что нам доступен большой неразмеченный корпус текстов $\mathcal{D}_U = \{(x_i)\}^N_{i=1}$ и небольшой размеченный валидационный набор $\mathcal{D}_{\text{dev}} = \{(x_j, y_j)\}^M_{j=1}$, причём $M \ll N$. Валидационный набор используется исключительно для отбора и фильтрации шумных эвристик и не применяется напрямую для обучения итогового классификатора.

Для создания слабых (неточных) меток на корпусе $\mathcal{D}_U$ введём набор функций разметки $\Lambda = \{\lambda_k\}^K_{i=1}$, где каждая функция $\lambda_k : X \to \{0,1,\emptyset\}$, либо сопоставляет входному объекту неточную метку класса, либо воздерживается от разметки. Такие функции могут представлять собой простые правила на основе ключевых слов, регулярные выражения или более сложные семантические конструкции. В рассматриваемой постановке функции разметки LF генерируются путём запроса к большой языковой модели с использованием инструкций и примеров на естественном языке.

Каждый элемент выборки $x_i \in \mathcal{D}_U$ может быть размечен несколькими функциями разметки одновременно, в результате чего формируется матрица меток $L \in \{0, 1, \emptyset\}^{N \times K}$. Вероятностная модель меток $p(y_i \mid L_{i,:})$ агрегирует полученные шумные разметки в виде мягких (вероятностных) меток $\hat{y}_i \in [0, 1]$, на которых затем обучается классификатор f_θ.

Центральная задача состоит в том, чтобы, используя обратную связь от модели f_θ, итеративно улучшать набор функций разметки Λ таким образом, чтобы каждое новое поколение функций LF точнее отражало семантику задачи, обеспечивало большее покрытие выборки и уменьшало уровень шума. Данный подход лежит в основе предлагаемого нами фреймворка LGIWL, который превращает процесс проектирования функций разметки в адаптивный цикл обратной связи, управляемый ошибками обучаемой модели.

4. Методология: итеративное обучение со слабым контролем на основе LLM

Фреймворк LGIWL основан на идее использования больших языковых моделей не в роли непосредственного классификатора, а в качестве генератора семантических правил, встроенного в итеративный процесс обучения со слабым контролем. Целью такого подхода является создание тщательно подобранного набора функций разметки, позволяющих

формировать информативные вероятностные метки для больших неразмеченных наборов данных при минимальном участии человека.

Пусть имеется неразмеченный корпус текстов и небольшой размеченный валидационный набор, где $M \ll N$.

Валидационный набор разделяется на две части: $\mathcal{D}_{\text{dev}}^{\text{prompt}}$, используемую для выявления ошибок текущего классификатора, и $\mathcal{D}_{\text{dev}}^{\text{filter}}$, предназначенную для количественной оценки и фильтрации кандидатных функций разметки.

Каждая итерация $t \in \{1,...,T\}$ цикла LGIWL включает следующие шаги:

i. **Формирование подсказок (Prompt Construction).** Используя примеры из набора $\mathcal{D}_{\text{dev}}^{\text{prompt}}$, на которых текущий классификатор ошибается или демонстрирует неопределённость, формируется текстовая подсказка $\pi^{(t)}$. Примеры группируются по пакетам и передаются в большую языковую модель (например, модель класса GPT), которая генерирует набор кандидатных функций разметки в виде семантических правил, шаблонов или групп ключевых слов. Обозначим полученный набор как $\Lambda_{\text{cand}}^{(t)}$.

ii. **Нормализация и дедупликация (Normalization and Deduplication).** Далее, вновь обращаясь к языковой модели с набором $\Lambda_{\text{cand}}^{(t)}$, происходит слияние семантически эквивалентных правил и удаление зашумлённых или избыточных формулировок, в результате чего получается очищенный набор кандидатных функций.

iii. **Фильтрация на валидационных данных (Filtering on Development Data).** Каждая кандидатная функция разметки $\lambda_k \in \Lambda_{\text{cand}}^{(t)}$ оценивается на наборе $\mathcal{D}_{\text{dev}}^{\text{filter}}$. Функции, для которых показатели оценки точности (precision) и покрытия (coverage) превышают заданные пороги δ_p и δ_c соответственно, включаются в общий накопленный набор функций разметки $\Lambda^{(t)}$.

iv. **Разметка и агрегация (Labeling and Aggregation).** Все накопленные функции разметки из множества $\Lambda^{(t)}$ применяются к неразмеченному корпусу D*U* при помощи лёгкого движка правил (например, t-lite-it-1.0), что приводит к формированию матрицы меток:

$$L \in \{0,1,\emptyset\}^{N \times |\Lambda^{(t)}|}.$$

Затем генеративная модель меток ϕ (например, Snorkel) агрегирует полученные сигналы в вероятностные метки:

$$\hat{y}_i = \phi(L_{i,:}).$$

v. **Цикл обратной связи (Feedback Loop).** На полученных метках $\{(x_i, \hat{y}_i)\}$ обучается временный рабочий классификатор. Ошибки классификации на наборе $\mathcal{D}_{\text{dev}}^{\text{prompt}}$ фиксируются и используются для формирования подсказки следующей итерации $\pi^{(t+1)}$.

Итерации продолжаются до тех пор, пока качество слабых меток на наборе $\mathcal{D}_{\text{dev}}^{\text{filter}}$ не стабилизируется. После достижения стабилизации производится обучение финальной прогнозной модели на полученных слабых метках $\{(x_i, \hat{y}_i)\}$. Таким образом, высокопроизводительный классификатор оказывается полностью отделён от цикла проектирования функций разметки.

Разделяя стадии генерации правил, их применения и финального обучения, метод LGIWL сочетает семантическую гибкость языковых моделей с эффективностью и интерпретируемостью систем, основанных на правилах.

5. Экспериментальная часть

5.1 Задача и набор данных

Мы оцениваем предложенный подход на задаче бинарной текстовой классификации обращений клиентов в службу поддержки на русском языке. Цель задачи – определить, относится ли многоходовой диалог к вопросам, связанным с тарифами. Полный набор данных содержит 10,000 неразмеченных диалогов, собранных с платформы цифрового банкинга. Целевой класс представлен достаточно редко и составляет примерно 8.2%

Для разработки и оценки метода мы дополнительно разметили два подмножества данных. Валидационный набор из 1,000 размеченных диалогов используется в процессе обучения исключительно для генерации и фильтрации функций разметки, но не применяется напрямую для обучения итоговой модели. В рамках предлагаемого фреймворка LGIWL этот набор делится на две равные части: первая используется для формирования подсказок языковой модели на основе ошибок текущего классификатора, а вторая – для оценки качества кандидатных функций разметки перед их включением в общий набор LF.

Отдельный тестовый набор из тысячи диалогов с ручной разметкой оставлен для финальной оценки. Метки из тестового набора не используются при обучении или подборе моделей ни в одном из рассматриваемых подходов со слабым контролем.

5.2 Обзор сравниваемых методов

В экспериментальном сравнении мы используем следующие методы, отражающие различные сценарии обучения (см. табл. 1):

- **Zero-Shot Prompting (LLM).** Базовый zero-shot подход, в котором модель GPT-4 напрямую классифицирует диалоги на тестовом наборе, получая описание задачи и несколько примеров (zero-shot). Данный подход представляет собой классификацию без явного обучения на размеченных данных.
- **Snorkel (Keywords).** Классический подход обучения со слабым контролем, при котором функции разметки создаются вручную или автоматически на основе TF–IDF ассоциаций с целевым классом. Полученные функции агрегируются с помощью Snorkel для генерации вероятностных меток.
- **CatBoost + Snorkel (Keywords).** Дискриминативный классификатор, обученный на слабых метках, полученных с помощью описанного выше подхода Snorkel (Keywords). Этот метод демонстрирует прирост качества от применения дискриминативной модели поверх статических слабых меток.
- **CatBoost (Supervised, Small Data).** Классификатор на основе градиентного бустинга CatBoost, обученный напрямую на 1,000 размеченных примерах из валидационного набора. Этот подход отражает реалистичную ситуацию, когда небольшая ручная разметка данных доступна, и позволяет оценить достаточность простой супервизии в условиях ограниченных данных.
- **Snorkel (Default LLM).** Функции разметки генерируются с помощью общей языковой модели по определению целевого класса и инструкциям, затем агрегируются через Snorkel. Данный подход не использует итеративную обратную связь и представляет собой одношаговый вариант разметки с помощью LLM с базовым заданным вручную описанием целевого класса.
- **LGIWL (CatBoost и RuModernBERT).** Предлагаемый нами метод, комбинирующий ключевые слова, функции разметки на основе LLM и итеративную доработку, основанную на обратной связи от классификатора. Новые LF итеративно

генерируются, оцениваются по точности и покрытию на валидационном наборе и добавляются в агрегируемый набор функций. Итоговые слабые метки используются для обучения классификатора CatBoost или тонко настроенного трансформера RuModernBERT. Этот метод отражает полный подход к адаптивной, масштабируемой разметке.

Табл. 1. Сводка сравниваемых методов и источников разметки.
Table 1. Summary of Compared Methods and Annotation Sources.

Метод	Описание	Источник разметки
Zero-Shot Prompting (LLM)	Прямая Zero-shot классификация с помощью LLM	Только промпты
Snorkel (Keywords)	Статические функции разметки на основе ключевых слов, агрегированные через Snorkel	Эвристика (TF–IDF)
CatBoost + Snorkel (Keywords)	CatBoost, обученный на слабых метках от Snorkel (Keywords)	Слабый контроль (статический)
CatBoost (Supervised, Small Data)	CatBoost, обученный на 1000 размеченных примерах	Ручная разметка (ограниченная)
Snorkel (Default LLM)	LFs, сгенерированные из LLM и агрегированные через Snorkel	Слабый контроль (LLM)
LGIWL (CatBoost / RuModernBERT)	Итеративная генерация LF через LLM, фильтрация и обучение	Слабый контроль (адаптивный)

5.3 Детали реализации

Все сравниваемые методы реализованы с использованием единой инфраструктуры и единых процедур обучения. В качестве дискриминативных моделей используются CatBoost (градиентный бустинг, эффективный на небольших структурированных данных) и RuModernBERT-base (предобученная трансформерная модель для русского языка). Слабые метки генерируются через Snorkel и используются как мягкие целевые метки для обучения моделей. При этом размеченные вручную метки во время обучения дискриминативных моделей не используются.

Функции разметки создаются как комбинация ключевых слов, семантических шаблонов и условий, сгенерированных с помощью LLM. В методе LGIWL новые LF итеративно предлагаются языковой моделью по ошибочным примерам и принимаются в общий набор только в случае превышения порогов покрытия и точности, оцениваемых на выделенной части валидационного набора.

5.4 Протокол оценки

Результаты классификации на тестовом наборе оцениваются по пяти метрикам: площадь под ROC-кривой (AUC), средняя точность (AP), точность (precision), полнота (recall) и F1-мера. Пороги классификации выбираются по максимуму F1-меры на валидационном наборе. Помимо точности классификации, мы анализируем покрытие и разнообразие правил, калибровку моделей и эффективность использования LLM.

6. Результаты и анализ

В данном разделе представлены результаты сравнительного анализа предложенного фреймворка LGIWL и базовых методов, описанных в разделе 5. Оценка проводилась на тестовом наборе из тысячи вручную размеченных диалогов с использованием стандартных метрик качества классификации: площади под ROC-кривой (AUC), средней точности (AP), точности (precision), полноты (recall) и F1-меры. Пороги для классификации подбирались по максимуму F1-меры на валидационном наборе и опущены в описании для удобства чтения.

6.1 Основные результаты

В табл. 2 представлены результаты оценки всех рассматриваемых методов. LGIWL, особенно в реализации на основе RuModernBERT, демонстрирует наилучшие результаты по всем метрикам, превосходя как традиционные подходы со слабым контролем, так и zero-shot классификацию с помощью LLM. Также LGIWL превосходит супервизированную модель CatBoost, обученную на тысяче размеченных примерах.

6.2 Сравнение супервизированного обучения и адаптивного слабого контроля

Хотя супервизированная модель CatBoost, обученная на 1,000 вручную размеченных примерах, показывает стабильные результаты (F1-мера: 0.816), она не превосходит метод LGIWL. Важно подчеркнуть, что LGIWL достигает более высоких результатов, не используя напрямую вручную размеченные данные при обучении, а расходуя аналогичный объём ручной разметки лишь на фильтрацию правил и уточнение подсказок. Это демонстрирует преимущество адаптивного обучения со слабым контролем по сравнению с прямым супервизированным обучением при ограниченном бюджете разметки.

Табл. 2. Сравнение качества моделей на тестовом наборе.
Table 2: Comparison of Model Performance on the Test Set.

Метод	AUC	AP	Точность	Полнота	F1-мера
Zero-Shot Prompting (LLM)	0,926	0,612	0,677	0,890	0,769
Snorkel (Keywords)	0,975	0,821	0,765	0,712	0,738
CatBoost + Snorkel (Keywords)	0,979	0,828	0,836	0,699	0,761
Snorkel (Default LLM)	0,732	0,323	0,800	0,219	0,344
CatBoost (ручная разметка, 1000 примеров)	0,977	0,877	0,800	0,833	0,816
LGIWL + CatBoost	0,982	0,905	0,851	0,792	0,820
LGIWL + RuModernBERT	**0,987**	**0,913**	**0,908**	**0,822**	**0,863**

6.3 Повышение точности как основной фактор роста F1-меры

Ключевым фактором более высокой F1-меры при использовании метода LGIWL является существенное улучшение точности по сравнению с другими подходами. Например, при запуске модели LGIWL на нейронной сети RuModernBERT достигается точность 0.908 при сохранении полноты 0.822. Для сравнения, метод запросов без явного обучения обеспечивает высокую полноту (0.890), но значительно проигрывает в точности (0.677), ограничивая тем самым итоговую F1-меру на уровне 0.769. Это указывает на то, что метод LGIWL не только

эффективно выявляет истинные положительные примеры, но и успешно избегает переизбыточной разметки, благодаря более точным и семантически осмысленным правилам.

6.4 Запросы без явного обучения: высокая полнота, низкая эффективность

Метод запросов без явного обучения (zero-shot prompting) на основе нейронной сети GPT-4 показывает относительно высокую полноту, однако требует значительных трудозатрат на ручную настройку. Разработка качественных подсказок потребовала множества итераций с участием человека, а также специфической настройки под задачу. Несмотря на отсутствие явного обучения, данный подход требует значительных ресурсов человека и вычислительных затрат, при этом обеспечивая лишь среднее итоговое качество. Его низкая точность и ограниченная адаптивность значительно проигрывают автоматизированному, основанному на обратной связи процессу LGIWL.

6.5 Эффект от итеративного уточнения функций разметки

Сравнение одношагового подхода Snorkel (Default LLM) с методом LGIWL иллюстрирует преимущество генерации подсказок по ошибкам классификатора. Функции разметки, сгенерированные за один шаг моделью LLM, показывают значение F1-меры, равное 0.344, тогда как итеративное уточнение правил в LGIWL повышает этот показатель до 0.820 (на модели CatBoost) и 0.863 (на сети RuModernBERT). Улучшение обусловлено не просто количеством правил, а их целенаправленной настройкой под ошибки классификации.

6.6 Ограничения статической разметки по ключевым словам

Статические методы на основе ключевых слов, включая Snorkel (Keywords) и CatBoost на Snorkel (Keywords), уступают методу LGIWL. Даже с использованием дискриминативного классификатора поверх слабых меток, лучшая достигнутая F1-мера не превышает 0.761. Подобные эвристики плохо справляются с перефразированием, многоходовыми контекстами и отрицаниями – ситуациями, которые LGIWL успешно обрабатывает за счёт семантической генерации правил через LLM.

6.7 Выводы

Метод LGIWL представляет собой надёжный и экономичный подход к обучению со слабым контролем. Он превосходит традиционные эвристики и методы запросов без явного обучения LLM по точности, полноте и F1-мере. Итеративная структура с обратной связью позволяет целенаправленно синтезировать и отбирать правила, улучшая обобщающую способность при сниженных затратах. Благодаря высокой точности и полноте, LGIWL обладает значимыми преимуществами для задач реальной классификации, где важны как корректность, так и полнота разметки.

7. Заключение

В данной работе мы представили LGIWL – итеративный подход к обучению со слабым контролем, основанный на генерации, уточнении и фильтрации функций разметки с помощью больших языковых моделей (LLM). Предложенный метод позволяет добиться высокого качества классификации в условиях ограниченного объёма размеченных данных, не прибегая к масштабной ручной аннотации. В отличие от подходов без явного обучения, требующих экспертной настройки инструкций и при этом страдающих от недостаточной точности, метод LGIWL использует большие языковые модели не напрямую как классификаторы, а как генераторы семантических правил, интегрированных в итеративный цикл с обратной связью по результатам работы классификатора.

Эксперименты на реальной задаче классификации банковских диалогов показали, что LGIWL существенно превосходит статические подходы обучения со слабым контролем и методы без явного обучения, достигая F1-меры 0,863 в реализации на основе сети RuModernBERT при высокой полноте и заметном росте точности. Полученные результаты сопоставимы с полностью супервизированными моделями, обученными на таком же количестве размеченных данных, и в некоторых случаях превосходят их. Это подтверждает, что стратегическое использование ограниченного набора размеченных данных для уточнения правил может быть эффективнее прямого обучения модели.

В отличие от полностью ручных подходов, таких как подбор ключевых слов или разработка подсказок для языковых моделей, LGIWL требует минимальной аннотации, направленной исключительно на фильтрацию функций разметки по ошибкам классификатора. Метод автоматизирует выявление выразительных и высокопокрывающих правил, многие из которых отражают нетривиальные семантические закономерности, трудные для ручного кодирования.

В будущем планируется расширение LGIWL на многоклассовые задачи и сценарии междоменного переноса знаний, а также интеграция более продвинутых стратегий генерации подсказок (например, цепочек рассуждений и саморефлексии) в цикл создания функций разметки. Также планируется формализация критериев сходимости при итеративном слабом контроле с использованием LLM и изучение возможностей интеграции сигналов активного обучения в контур обратной связи.

Таким образом, LGIWL представляет собой масштабируемую, интерпретируемую и экономичную альтернативу как ручной разметке данных, так и непрозрачным zero-shot подходам, двигаясь в направлении автоматизированного, но управляемого человеком применения языковых моделей в реальных задачах машинного обучения.

Список литературы / References

[1]. Stephen H Bach, Ben He, Alexander Ratner, and Christopher Ré. Learning the structure of generative models without labeled data. In International Conference on Machine Learning, pages 273–282, 2017.
[2]. Stephen H Bach, Daniel Rodriguez, Yintao Liu, et al. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In ACM SIGMOD, pages 362–375, 2019.
[3]. Bradley Denham et al. Witan: Unsupervised labeling function generation for assisted data programming. Proceedings of the VLDB Endowment, 15(11): 2334–2347, 2022.
[4]. Nan Guan et al. Datasculpt: Cost-efficient label function design via prompting large language models. In EDBT, pages 226–237, 2025.
[5]. Tai-Hsuan Huang et al. Scriptoriumws: A code generation assistant for weak supervision. In ICLR Workshop, 2023. arXiv:2301.01229.
[6]. Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision.
[7]. Robert Smith et al. Language models in the loop: Incorporating prompting into weak supervision. Journal of Data Science, 1(2):1–30, 2022.
[8]. Paroma Varma and Christopher Ré. Snuba: Automating weak supervision to label training data. In VLDB Endowment, volume 12, pages 223–236, 2018.
[9]. Peng Yu and Stephen H Bach. Alfred: A system for prompted weak supervision. In ACL System Demonstrations, pages 479–488, 2023.
[10]. Jialu Zhang et al. Wrench: A comprehensive benchmark for weak supervision. NeurIPS Datasets and Benchmarks, 2021.
[11]. Ruixiang Zhang et al. Prboost: Prompt-based rule discovery and boosting for interactive weakly-supervised learning. In ACL, pages 745–758, 2022.

Информация об авторах / Information about authors

Артур Дмитриевич СОСНОВИКОВ – аспирант Института системного программирования с 2023 года. Сфера научных интересов: методы машинного обучения, обучение со слабым контролем.

Artur Dmitrievich SOSNOVIKOV – graduate student at the Institute of System Programming since 2023. Research interests: machine learning methods, weakly supervised learning.

Антон Дмитриевич ЗЕМЕРОВ – старший ML-инженер в банке «Точка». Выпускник Физтех-Школы Прикладной Математики и Информатики МФТИ. Сфера научных интересов: методы машинного обучения, обработка естественного языка, большие языковые модели.

Anton Dmitrievich ZEMEROV – Senior ML Engineer at Tochka Bank. Graduate of the PhysTech School of Applied Mathematics and Informatics at MIPT. Research interests: machine learning methods, natural language processing, large language models.

Денис Юрьевич ТУРДАКОВ – кандидат физико-математических наук, заведующий отделом ИСП РАН, доцент кафедры системного программирования факультета ВМК МГУ. Научные интересы: анализ естественного языка, извлечение информации, обработка больших данных, анализ социальных сетей.

Denis Yurievich TURDAKOV – Cand. Sci. (Phys.-Math.), Head of Department at ISP RAS, associate professor of the Department of System Programming at MSU. Research interests: natural language processing, information extraction, big data analysis, social network analysis.

DOI: 10.15514/ISPRAS-2025-37(6)-21

Beyond LLVM: Evaluating Fast Code Generation Alternatives for Query Compilation in PostgreSQL

M.V. Pantilimonov, ORCID: 0000-0003-2277-7155 <pantlimon@ispras.ru>
R.A. Buchatskiy, ORCID: 0000-0001-8522-1811 <ruben@ispras.ru>
D.V. Zavedeev, ORCID: 0009-0009-1477-5249 <zdenis@ispras.ru>

Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. The evolution of query compilation in database management systems traces back to System R, which pioneered a code generation scheme where small machine code fragments were stitched together to form a specialized routine to process a given SQL statement. Subsequent approaches shifted to generating C code, compiling it with system compilers like GCC into dynamic libraries, and loading them at runtime. The current state-of-the-art standard for dynamic query compilation is the LLVM framework, which bypasses frontend compiler overhead by directly generating intermediate representation, enabling machine-independent optimizations and efficient machine code generation. LLVM's resource-intensive nature, primarily designed as an optimizing compiler, however, can lead to compilation times that are orders of magnitude longer than query execution times, particularly problematic for queries with millisecond-level interpretation costs. This paper evaluates two lightweight code generation frameworks for x86-64 architecture as alternatives to LLVM in PostgreSQL, assessing their code generation speed and the quality of emitted machine code. We present a qualitative comparison with LLVM, analyzing trade-offs between compilation latency and runtime performance across databases of varying sizes. Experimental results demonstrate that lightweight code generation can not only outperform LLVM on small-scale datasets but also maintain competitive performance on larger ones.

Keywords: dynamic compilation; JIT-compilation; query execution; DBMS; PostgreSQL; LLVM; DynASM; AsmJIT.

For citation: Pantilimonov M.V., Buchatskiy R.A., Zavedeev D.V. Beyond LLVM: Evaluating Fast Code Generation Alternatives for Query Compilation in PostgreSQL. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 77-92. DOI: 10.15514/ISPRAS-2025-37(6)-21.

Не LLVM единым: Исследование альтернативных методов быстрой генерации кода для компиляции запросов в PostgreSQL

М.В. Пантилимонов, ORCID: 0000-0003-2277-7155 <pantlimon@*ispras.ru*>
Р.А. Бучацкий, ORCID: 0000-0001-8522-1811 <ruben@ispras.ru>
Д.В. Заведеев, ORCID: 0009-0009-1477-5249 <zdenis@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. Идея компиляции запросов в системах управления базами данных берёт своё начало в System R, где впервые была реализована схема генерации кода, при которой небольшие фрагменты машинного кода объединялись вместе для создания специализированной подпрограммы, обрабатывающей конкретный SQL запрос. В дальнейшем подходы изменились: вместо машинного кода начали генерировать код на языке C, который затем компилировался с помощью системных компиляторов, таких как GCC, в динамические библиотеки и подгружался в процессе выполнения. Сегодня стандартом де-факто в области динамической компиляции запросов стал фреймворк LLVM. Благодаря своей модульной архитектуре он позволяет избежать дорогостоящего этапа трансляции с языка высоко уровня в промежуточное представление, обеспечивая его прямую генерацию с последующим применением машинно-независимых оптимизаций и генерации эффективного машинного кода. Однако LLVM изначально разрабатывался как оптимизирующий компилятор, и его использование может приводить к значительным накладным расходам на компиляцию – в отдельных случаях они превышают время выполнения запроса в десятки раз. Это особенно проблематично для коротких запросов с миллисекундным временем исполнения. В данной работе рассматриваются два легковесных генератора кода для архитектуры x86-64 в качестве альтернативы LLVM в СУБД PostgreSQL. Оцениваются как скорость генерации кода с использованием этих фреймворков, так и качество получаемого исполняемого кода. Приведено качественное сравнение с LLVM, анализируются компромиссы между скоростью компиляции и производительностью выполнения запросов на базах данных различного размера. Результаты экспериментов показывают, что легковесные решения не только превосходят LLVM по производительности на небольших наборах данных, но и сохраняют её конкурентноспособной на больших объёмах информации.

Ключевые слова: динамическая компиляция; JIT-компиляция; выполнение запросов; СУБД; база данных PostgreSQL; платформа LLVM; компилятор DynASM; компилятор AsmJIT.

Для цитирования: Пантилимонов М.В., Бучацкий Р.А., Заведеев Д.В. Не LLVM единым: Исследование альтернативных методов быстрой генерации кода для компиляции запросов в PostgreSQL. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 77–92 (на английском языке). DOI: 10.15514/ISPRAS–2025–37(6)–21.

1. Introduction

Modern database management systems (DBMS) process queries through a multi-stage pipeline, beginning with the translation of a declarative query into a logical query plan represented as a tree of relational algebra operators. This logical plan is then converted into physical execution plan, which specifies data access methods and join algorithms. PostgreSQL DBMS [1] is a textbook example of this architecture, using the well-known iterator model, a.k.a. Volcano model [2], where query execution follows a pull-based approach – data flows from leaf nodes upward to the root, with each operator recursively requesting tuples from its children. While newer push-based execution model, popularized by Hyper [3], has demonstrated better performance – even when retrofitted onto Volcano-based execution engine not originally designed for it [4] – adopting such an approach would require a fundamental redesign of both the execution engine and the query planner. This transition would also introduce particular complexities in handling of certain operations like limit and merge joins [5]. Consequently, PostgreSQL continues to rely on iterator-based model, though substantial optimizations in expression evaluation have been implemented over the years. Version 10 introduced a bytecode-based interpreter [6-8] with direct threading technique that minimizes

dispatch overhead while improving branch prediction and cache locality. Building upon this foundation, PostgreSQL 11 added LLVM-based JIT [9] compilation for expressions [10], further accelerating analytical workloads. However, JIT compilation does not come for free – while it improves execution speed, the compilation overhead might outweigh these performance gains. This is particularly problematic with LLVM, which is fundamentally an optimizing compiler rather than a lightweight JIT engine. Its optimization pipeline and machine code emission are inherently expensive [11], making it almost certain to degrade performance when processing small datasets – cases where interpretation would be faster.

The most straightforward approach to decide whether to apply JIT compilation is to rely on the DBMS planner's cost estimation before query execution. However, even with perfect statistics and the most accurate cost models, these estimations will inevitably be wrong for some queries. Every such misprediction forces the system to pay full compilation costs for queries that would have executed faster via interpretation. To address this, multi-tier JIT compilation offers a promising direction, adopted in browser engines and virtual machines to balance startup latency and peak performance. Prior research [12] has explored system design with multi-tier compilation in mind, starting with interpretation and later switch to optimized LLVM-generated code after reaching a certain execution threshold. Alternatively, interpretation can be bypassed entirely, generating unoptimized machine code first [13] and then transitioning to optimized LLVM code compiled in parallel.

While such design performs well on server-grade hardware with abundant CPU cores, it becomes less optimal on commodity hardware. In more modest systems, dedicating resources to background compilation may actually reduce overall throughput. These computational resources could instead be used to serve additional concurrent queries, prioritizing system-wide throughput over single-query latency. The issue with (multi-tier) JIT in database systems lies in the low reusability of compiled query code, in contrast to browser engines and VMs. While one could implement a compiled query cache, the bookkeeping overhead is far from negligible and introduces significant challenges:

- Cache management: deciding which queries to cache, when to evict them and how to efficiently retrieve them. Cache invalidation may occur due to statistics updates or schema changes.
- Key selection: determining the cache key – should it be based on the query text, AST, execution plan, IR, etc. The key generation process is computationally expensive regardless of the chosen approach.
- Handling constants: optimize code with literals i.e. "colA > 5" or replace literal with variable i.e. "colA > :x" to reuse compiled queries for different values. This is a trade-off between the number of compiled queries and plan quality due to varying value distributions.
- Handling memory addresses: absolute addresses used in compiled code must be replaced with relative addresses or patched before execution. This introduces either extra metadata to track address locations needing patching along with a patching step or requires careful memory management and relative address calculations.

In contrast, JIT-compiled code in VMs or browser engines executes naturally when control flow reaches it, requiring minimal bookkeeping. Depending on compilation latency, it may be simpler and more efficient to compile queries on-demand each time rather than managing a query compilation cache.

We believe that the optimal solution lies in a JIT compiler with extremely low compilation latency, ensuring it almost never increases overall query execution time. For the simplest queries, however, interpretation should still be preferred, as it remains faster than even the most minimal compilation pipeline.

In our research, the overall goal is to assess whether lightweight JIT compilers can achieve performance comparable to LLVM-based JIT compilers in DBMS environments, without

introducing excessive compilation latency due to inaccurate query cost estimation. PostgreSQL serves as an ideal candidate for this experiment, as it already has a well-integrated LLVM-based JIT compiler, which is considered the industry standard. Additionally, PostgreSQL provides a simple API for integrating alternative JIT compiler. The scope of JIT compilation required for the TPC-H [14] benchmark is minimal: only two functions need to be compiled – tuple deforming and expression evaluation. Notably, expression evaluation does not require support for all operation types to handle TPC-H benchmark completely.

2. Taxonomy of JIT compilers

JIT compilers can be broadly classified into three categories based on their design philosophy, optimization capabilities and compilation overhead: heavyweight, medium-weight and lightweight.

Heavyweight JIT compilers are characterized by their extensive intermediate representations and long machine-independent optimization pipeline. Examples include LLVM and GCC [15], which are originally designed as optimizing AOT (Ahead-of-Time) compilers but can also be used as JIT compilers. This category also includes JIT compilers tightly coupled with specific run-time environments, tailored to particular object models, garbage collectors or bytecode peculiarities. Notable examples include V8 [16], C1/C2 (HotSpot) [17], RyuJIT (.NET) [18], Parrot (Perl) [19] and MoarVM (Raku) [20]. While these compilers typically generate highly optimized code, they either incur significant compilation latency or are difficult/impossible to integrate into systems outside their native run-time environments.

Medium-weight JIT compilers use a lightweight intermediate representation with reduced set of optimizations. Following the 80/20 principle, they aim to deliver 80% of GCC -O2 performance level with only 20% of the code. Due to their resemblance to scaled-down versions of LLVM, these compilers are sometimes referred to as “mini-LLVM”. A notable example is GNU LibJIT [21].

Lightweight JIT compilers lack a high-level IR and perform no machine-independent optimizations. At most, they may use a very low-level IR, effectively acting as platform-independent assemblers. These compilers prioritize fast code generation and emit machine code with minimal overhead. Examples include DynASM [22], AsmJIT [23], GNU lighting [24], sljit [25], xbyak [26], etc.

3. Lightweight JIT compilers

In the current phase of the research, we explored two lightweight JIT compilers: AsmJIT and DynASM. AsmJIT is a mature project with industrial applications that offers an abstraction with automatic register allocation – an interesting feature for comparison against manual approach in the same task. DynASM, on the other hand, promises outstanding code generation performance due to its preprocessing step and is part of widely used LuaJIT compiler for the Lua programming language.

3.1 DynASM

DynASM is a dynamic assembler designed for code generation engines and was originally developed as part of the LuaJIT project. It was initially created to serve as x86 JIT compiler in LuaJIT 1.1 (last released in 2010). In the current LuaJIT 2.1 version, DynASM is primarily used to generate the interpreter core, which is written in hand-optimized assembly. LuaJIT 2.0 introduced a new trace-based JIT compiler [27] with SSA-based intermediate representation along with numerous optimizations. This version also replaced DynASM with a new machine code generator for better performance in the JIT compiler. DynASM preprocesses source file with a mixture of C/C++ and assembly code and produces modified output file. The assembler directives are replaced with runtime calls, primarily “dasm_put(...)”, while the corresponding assembly instructions are embedded in an “actionlist” array in internal format. This array contains both the raw instructions and opcode-like metadata that defines how the code should be dynamically patched with actual values during runtime code generation phase. This micro-template-based approach considerably reduces code emission overhead. DynASM’s minimal runtime library handles code generation,

relocation and linking by processing the precompiled “actionlist”. As LuaJIT 2.1 maintains x86, x64, ARM, ARM64, PPC and MIPS architecture, DynASM correspondingly supports these platforms to generate the assembly-written interpreter core for each target architecture.

3.2 AsmJIT

AsmJIT is a lightweight, high-performance C++ library for dynamic code generation, supporting x86, x64, AArch64 architectures, with AArch32 support currently in development. Its well-thought architecture offers multiple multiple layers of abstraction, each providing varying levels of manual control to suit different use cases:

1. Assembler – The lowest-level interface, offering minimal abstraction over emitted instructions. It directly emits machine code into code buffer with no further transformations.
2. Builder – Built on top of Assembler, this layer provides simple code representation, which wraps each instruction into a node within a double-linked list. This structure enables flexible instruction rearrangement and basic code analysis, i.e. register usage tracking. The latter can be useful to optimize controlled call sites, reducing register pressure by neglecting strict calling convention to some extent.
3. Compiler – The highest-level abstraction, extending the Builder interface with virtual registers, automated register allocation, and ABI compliance for function calls. For register allocation it uses linear-scan algorithm with live-range analysis, simplifying a code generator implementation.

Layered design enables applications to combine different emitters as needed – manual register management for hot paths and automatic allocation for less critical code. AsmJIT is a well-established project that has high recognition in academic and several industrial level applications, e.g., Erlang JIT compiler [28].

4. Evaluation

4.1 Scope of JIT compilation

For evaluation purposes, we have selected PostgreSQL 17.2 as a candidate for experimentation. This DBMS follows a classical query processing pipeline (Fig. 1) that starts with lexical and syntactic analysis to parse client input and produce a parse tree. The parse tree then undergoes semantic analysis, where it is augmented with system and schema metadata to produce a query tree. Subsequently, a series of rewrite rules transform and canonicalize the query tree. Finally, planning and optimization convert the query tree into the most cost-effective logical plan, which is then translated into a physical plan suitable for execution via interpretation.

The PostgreSQL interpreter can be roughly divided into two parts. The first one implements data access, join operations and auxiliary operators such as Limit, Order by, Union (All). The second handles expression evaluation. These two parts are tightly integrated. For instance, during most join operations, the expression engine is invoked to compare attribute values between tuples. The expression engine uses a bytecode-like approach (Fig. 2), where the selected version implements 100 distinct opcodes, each representing a specific operation. Instruction flow between opcodes is managed via a dispatch table – provided the compiler used to build the DBMS binary supports computed gotos. This table contains addresses pointing to the relevant code block for each opcode. If computed gotos are unsupported, a standard switch statement is used as a fallback mechanism.

To achieve full compilation support for all expressions in the TPC-H benchmark, code generation must be implemented for a subset of 35 opcodes. Among these, one of the most performance-critical operations is tuple deforming, which converts on-disk tuple representation into in-memory format. This operation can be accelerated by creating a function specific to the table layout and the set of

columns being accessed. To ensure a proper comparison across different JIT implementations, all alternative providers must support compilation of this defined subset of expression engine, including the tuple deforming operation.

To implement code generation for required opcodes, the JIT compiler only needs to support a minimal set of operations – basic arithmetic and bitwise operations (mainly for computing offsets), memory loads/stores, branch instructions with labels (to implement control flow), and label referencing (to efficiently lower switch statements into jump tables).

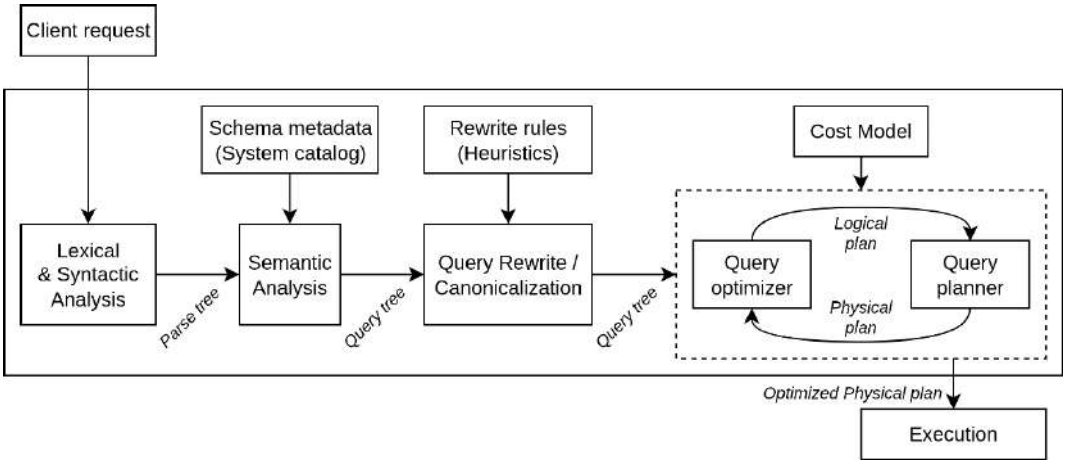


Fig 1. Classical query processing pipeline in DBMS.

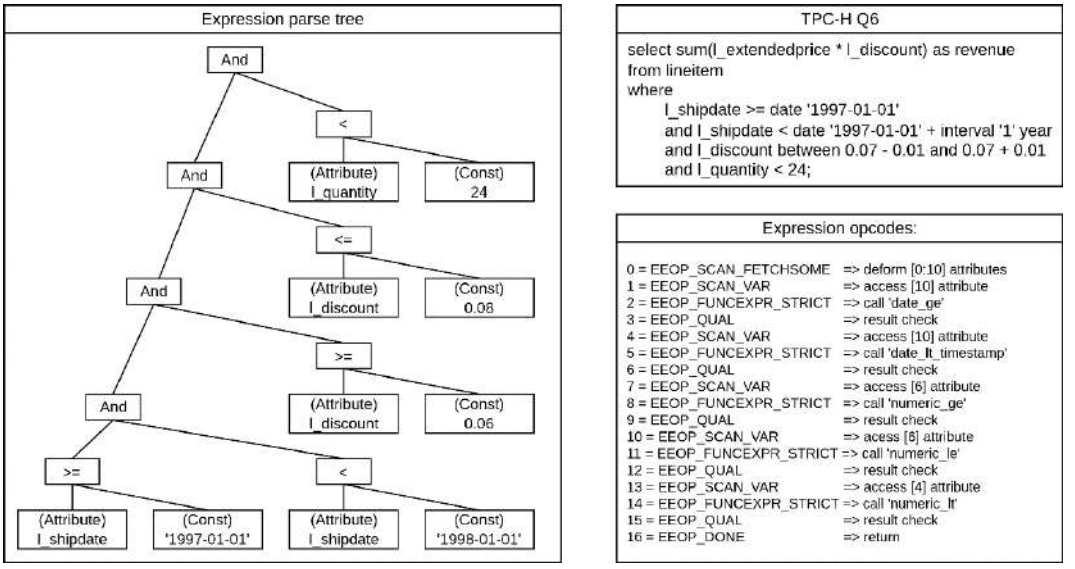


Fig 2. PostgreSQL expression opcodes for TPC-H Q6 WHERE clause.

4.2 Implementation fragment

We present a minimal excerpt from the implementation of lightweight JIT compilers (Fig. 3). In the case of AsmJIT, additional code wrappers are used to simplify field offset calculations and ensure that load and store operations match the specified data types. DynASM, on the other hand, uses preprocessor-level directives for type declarations, which automatically generate the required field

offsets. Overall, the process of writing JIT-compiled code is no more complex than writing traditional assembly by hand – and in the case of AsmJIT, it is even simpler due to its higher-level abstraction that automates register allocation.

```
bool asmjit_compile_expr(struct ExprState *state) {
  // ...
  for (int opno = 0; opno < state->steps_len; opno++) {
    // ...
    switch (opcode) {
    // ... MORE OPCODES
    case EEOP_INNER_VAR:
    case EEOP_OUTER_VAR:
    case EEOP_SCAN_VAR: {
      x86::Gp v_slot = opcode == EEOP_INNER_VAR   ? v_innerslot
                     : opcode == EEOP_OUTER_VAR ? v_outerslot
                                                : v_scanslot;
      const auto attnum = op->d.var.attnum;
      x86::Gp v_resvalue = cc::load_const_voidptr(
        jcc, "v_resvalue[attnum]", op->resvalue);
      x86::Gp v_values = ccTupleTableSlot::load_tts_values(jcc, v_slot);
      x86::Gp v_value = cc::load_array_at<Datum>(jcc, v_values, attnum);
      cc::store_at(jcc, v_resvalue, v_value);

      x86::Gp v_resnull = cc::load_const_voidptr(
        jcc, "v_resnull[attnum]", op->resnull);
      x86::Gp v_nulls = ccTupleTableSlot::load_tts_isnull(jcc, v_slot);
      x86::Gp v_isnull = cc::load_array_at<bool>(jcc, v_nulls, attnum);
      cc::store_at(jcc, v_resnull, v_isnull);
      break;
    }
    }
  }
  // ...
}
```

```
bool dynasm_compile_expr(ExprState *state) {
  // ...
  for (int opno = 0; opno < state->steps_len; opno++) {
    // ...
    switch (opcode) {
    // ... MORE OPCODES
    case EEOP_INNER_VAR:
    case EEOP_OUTER_VAR:
    case EEOP_SCAN_VAR: {
      const int slot_register =
        opcode == EEOP_INNER_VAR    ? R_INNERSLOT
        : opcode == EEOP_OUTER_VAR ? R_OUTERSLOT
                                   : R_SCANSLOT;
      | mov64 T1, (uintptr_t) op->resvalue
      | mov T0, t_slot:v_slot->tts_values
      | mov T0, [T0+DATUM_OFF(op->d.var.attnum)]
      | mov [T1], T0
      |
      | mov64 T2, (uintptr_t) op->resnull
      | mov T0, t_slot:v_slot->tts_isnull
      | mov T0b, [T0+BOOL_OFF(op->d.var.attnum)]
      | mov [T2], T0b
      ;
      break;
    }
    }
  }
  // ...
}
```

Fig 3. Excerpt from the implementation of lightweight JIT compilers.

4.3 Performance evaluation of JIT compilers

To evaluate performance of JIT compilers in PostgreSQL, we generated a set of TPC-H databases at varying factors (Fig. 4), starting from extremely low scale factor (SF) = 0.01 up to SF = 20 – ensuring that the entire database could fit withing the Linux page cache of the testing system. These varying scale factors allow us to examine how different levels of code generation and optimization impact query execution performance across different dataset sizes.

Scale factor (SF)	0.01	0.1	0.5	1	2.5	5	7.5	10	15	20
Database Size	109M	372M	1.7G	3.2G	6.2G	20G	24G	29G	40G	50G

Fig 4. TPC-H benchmark database size by scale factor.

The JIT capabilities in PostgreSQL are controlled through several configuration flags, which vary depending on the underlying engine: LLVM, AsmJIT, or DynASM. For testing, we selected the following configurations that control different aspects of code generation and optimization:

- {engine}_E – code generation only for expression nodes without optimizations.
- {engine}_EO – code generation for expression nodes with optimizations applied.
- {engine}_ED – code generation for expression nodes and deform functions without optimizations.
- {engine}_EDO – extends ED with optimizations applied after code generation.

- {engine}_EDOI – enhances EDO with inlining of built-in functions' source code (e.g. date_ge, int4mul, int8div) into the generated module, enabling direct inlining during optimization rather than indirect calls via absolute address constants.

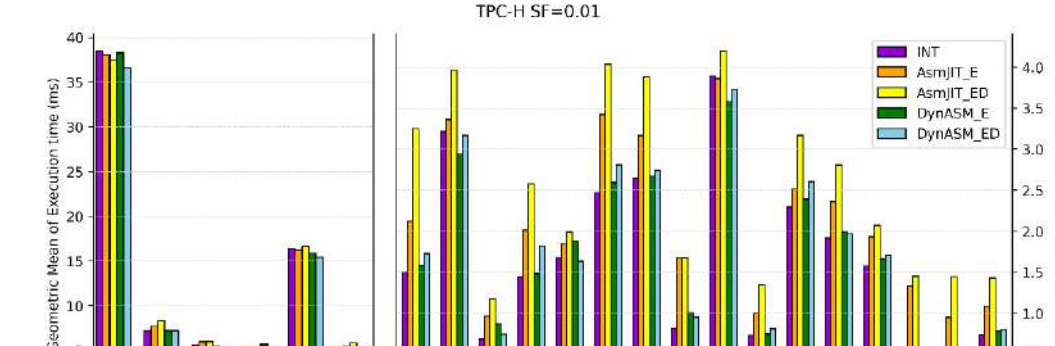
LLVM engine supports all five configurations, while AsmJIT and DynASM only use two – "{engine}_E" and "{engine}_ED" due to the absence of automatic optimizations and function inlining. Adding the latter would mean writing assembly generators by hand for each built-in function – a task that, while technically feasible, demands significant development effort.

All benchmarks were conducted on an Ubuntu 24.10 machine with an Intel i7-11700 CPU and 64GB DDR4 RAM (3200MHz). PostgreSQL 17.2 was built with GCC 14.2.0 and LLVM 18.1.8. PostgreSQL was configured with *shared_buffers* set to 8GB, *work_mem* to 256MB, and *max_parallel_workers_per_gather* to 0 to disable parallel query execution. Before each scale factor run, the entire database was preloaded into Linux page cache to eliminate I/O overhead. Each query was executed 21 times, with the first warmup run excluded; the geometric mean of execution times was recorded. Unless otherwise noted, execution times include all stages of query processing – from parsing to execution, including JIT compilation.

The results are presented in two parts. First, we compare the performance of AsmJIT and DynASM, focusing on both their compilation times and query execution times. Next, we evaluate the lightweight JIT engines against LLVM, using the interpreter as a baseline to assess the overall effectiveness of JIT compilation. We also analyze how the query execution performance of different JIT engines varies across database sizes and attempt to identify the threshold at which heavyweight JIT compiler begins to outperform the lightweight alternatives.

4.3.1 AsmJIT vs DynASM

Before proceeding, it is important to note that our AsmJIT-based JIT implementation relies on its *Compiler* abstraction layer, which automatically manages both register allocation and function calls. While this abstraction simplifies development, it leads to less predictable code quality and increased compilation overhead. In contrast, DynASM-based implementation requires everything to be handled manually, which improves compilation time due to the absence of abstraction, but makes performance highly dependent on the effectiveness of manual register allocation.



	q01	q09	q10	q13	q18	q21
INT	38.47	7.21	5.58	5.28	16.38	5.09
AsmJIT_E	38.08	7.69	5.96	5.31	16.19	5.42
AsmJIT_ED	37.49	8.33	5.99	5.24	16.62	5.82
DynASM_E	38.27	7.25	5.45	5.7	15.91	5.05
DynASM_ED	36.56	7.25	4.83	4.82	15.42	4.92

	q02	q03	q04	q05	q06	q07	q08	q11	q12	q14	q15	q16	q17	q19	q20	q22
INT	1.5	3.21	0.69	1.44	1.68	2.47	2.65	0.82	3.89	0.73	2.29	1.92	1.58	0.31	0.39	0.74
AsmJIT_E	2.12	3.36	0.96	2.01	1.85	3.42	3.17	1.68	3.86	1.01	2.51	2.36	1.93	1.33	0.95	1.08
AsmJIT_ED	3.25	3.96	1.18	2.57	1.99	4.04	3.88	1.68	4.2	1.34	3.17	2.8	2.07	1.45	1.45	1.43
DynASM_E	1.58	2.94	0.87	1.49	1.87	2.59	2.67	1.01	3.58	0.75	2.39	2.0	1.66	0.39	0.49	0.78
DynASM_ED	1.73	3.17	0.74	1.82	1.63	2.81	2.74	0.95	3.73	0.81	2.6	1.97	1.71	0.38	0.49	0.81

Fig 5. Geometric mean of execution time of lightweight JIT engines on TPC-H SF=0.01. Green table cells indicate that the JIT query execution time is lower than that of the interpreter.

In the context of TPC-H benchmark with a very small scale factor = 0.01, the execution time comparison between AsmJIT and DynASM (as shown in Fig. 5) reveals several observations:

- DynASM_E configuration outperforms AsmJIT_E across all queries except for q6 and q13. We suspect that the difference in performance for q1 and q6 may be due to execution-time fluctuations, as the median execution time is lower (Fig. 6). For q13, DynASM_E is consistently slower than AsmJIT_E, likely due to lower-quality code – a subject for future analysis.
- DynASM_ED configuration consistently outperforms AsmJIT_ED across all queries and AsmJIT_E, except for q15.
- ASMJIT_E improves performance in queries q1, q12, and q18, while ASMJIT_ED shows improvement in q1 and q13.
- DynASM_E achieves performance gains in queries q1, q3, q10, q12, q18, and q21, whereas DynASM_ED demonstrates improvements across a broader set of queries: q1, q3, q6, q10, q12, q13, q18, and q21.
- When considering the geomean speedup against the interpreter across all queries – AsmJIT_E: 0.758x, AsmJIT_ED: 0.653x, DynASM_E: 0.947x, and DynASM_ED: 0.949x – it becomes evident that compiling the deform function using AsmJIT actually degrades performance compared to the interpreter. In contrast, DynASM_ED slightly improves overall performance compared to the DynASM_E configuration. Notably, even at this minimal scale factor, DynASM achieves nearly neutral performance degradation, which we attribute to its highly efficient code generation and emission process.

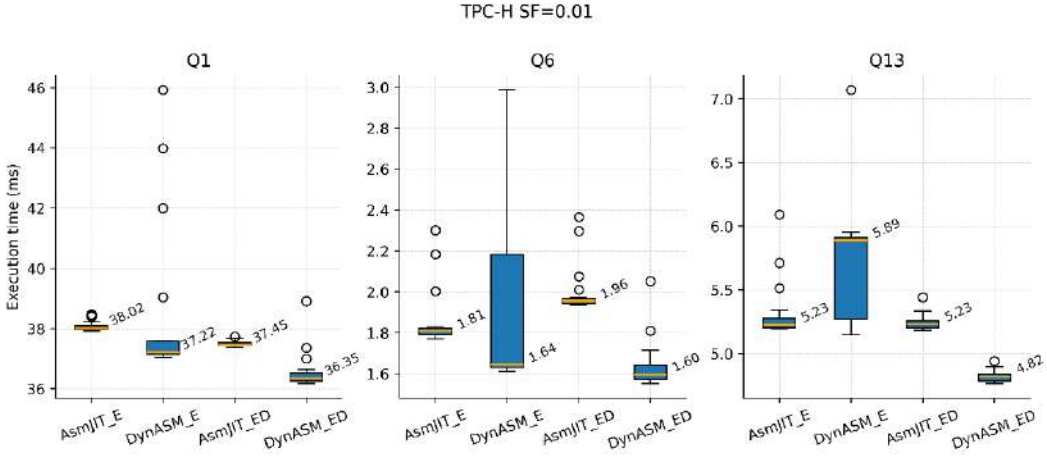


Fig. 6. Distribution of execution time of lightweight JIT engines on TPC-H SF=0.01 queries q1, q6, q13. The value near each box indicates the median execution time across 20 query runs.

Regarding compilation time (Fig. 7), that remains constant across all scale factors, we observe:

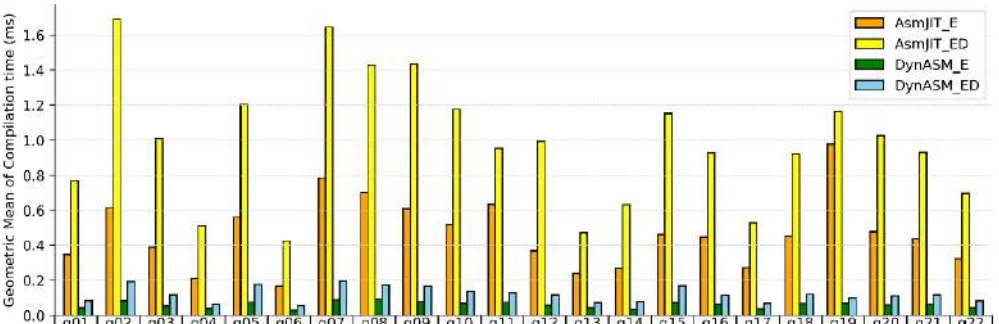
- DynASM_E achieves a geometric mean of compilation speedup of 7.5x over AsmJIT_E.
- DynASM_ED achieves a geometric mean of compilation speedup of 8.2x over AsmJIT_ED.

This shows that the DynASM-based implementation has significantly faster code generation and emission steps compared to AsmJIT-based implementation using its *Compiler* abstraction.

Finally, using AsmJIT_E as the baseline for code quality evaluation (Fig. 8), we observe the following:

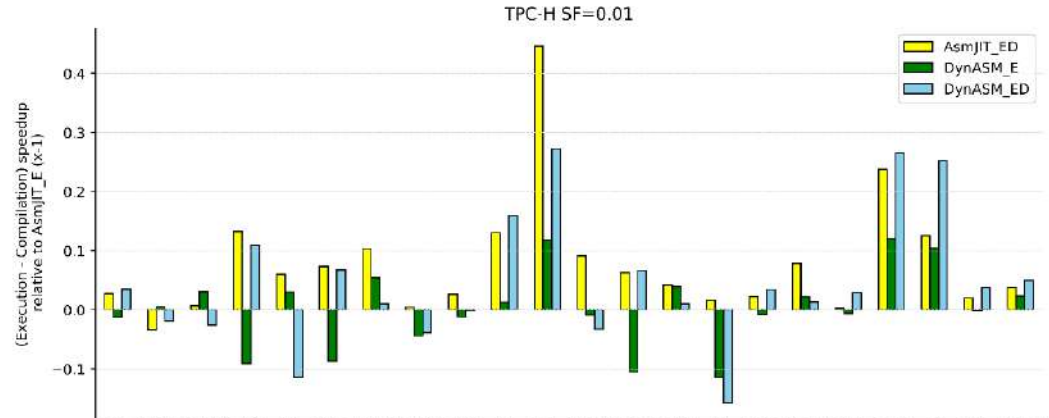
- DynASM_E produces lower-quality code in q1, q4, q6, q8, q9, q10, q12, q13, q15, q16, q18 and q21, with a peak regression of 0.88x measured in q15. Conversely, the maximum performance improvement is a 1.12x speedup, recorded in q19.

- The geometric mean speedup of DynASM_E is close to 1x, indicating that its overall code quality is comparable to AsmJIT_E.
- DynASM_ED exhibits a geometric mean speedup of 1.041x, while AsmJIT_ED achieves 1.074x – indicating that AsmJIT generates noticeably better code for the deform function, despite the hand-tuned assembly in the DynASM implementation, which avoids register spills entirely within the body of the function, with only minor spillage occurring in the prologue and epilogue.



	q01	q02	q03	q04	q05	q06	q07	q08	q09	q10	q11	q12	q13	q14	q15	q16	q17	q18	q19	q20	q21	q22
AsmJIT_E	0.35	0.61	0.39	0.21	0.56	0.17	0.78	0.7	0.61	0.52	0.63	0.37	0.24	0.27	0.46	0.45	0.27	0.45	0.98	0.48	0.44	0.32
AsmJIT_ED	0.77	1.69	1.01	0.51	1.2	0.42	1.64	1.43	1.43	1.17	0.95	0.99	0.47	0.63	1.15	0.93	0.53	0.92	1.16	1.03	0.93	0.7
DynASM_E	0.04	0.08	0.05	0.04	0.07	0.03	0.09	0.09	0.08	0.07	0.07	0.06	0.04	0.03	0.07	0.06	0.03	0.07	0.07	0.06	0.06	0.04
DynASM_ED	0.08	0.19	0.12	0.06	0.18	0.05	0.19	0.17	0.17	0.14	0.13	0.12	0.07	0.08	0.17	0.12	0.07	0.12	0.1	0.11	0.12	0.08

Fig 7. Compilation time of Lightweight JIT engines on TPC-H benchmark.



	q01	q02	q03	q04	q05	q06	q07	q08	q09	q10	q11	q12	q13	q14	q15	q16	q17	q18	q19	q20	q21	q22
AsmJIT_E	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
AsmJIT_ED	0.03	-0.03	0.01	0.13	0.06	0.07	0.1	0.0	0.03	0.13	0.45	0.09	0.06	0.04	0.02	0.02	0.08	0.0	0.24	0.13	0.02	0.04
DynASM_E	-0.01	0.0	0.03	-0.09	0.03	-0.09	0.05	-0.04	-0.01	0.01	0.12	-0.01	-0.1	0.04	-0.12	-0.01	0.02	-0.01	0.12	0.1	-0.0	0.02
DynASM_ED	0.03	-0.02	-0.03	0.11	-0.11	0.07	0.01	-0.04	-0.0	0.16	0.27	-0.03	0.07	0.01	-0.16	0.03	0.01	0.03	0.27	0.25	0.04	0.05

Fig 8. Execution - Compilation time speedup, relative to AsmJIT_E. The metric is defined as (Speedup - 1), where Speedup = (AsmJIT_E["Execution time"] – AsmJIT_E["Compilation time"]) / (Target ["Execution time"] - Target["Compilation time"]).
Both execution and compilation times represent geometric mean values based on 20 query runs.
Value of 0 corresponds to equal performance;
positive values denote speedups, and negative values denote slowdowns.

4.3.2 Lightweight against LLVM

We begin our comparison of lightweight JIT engines against LLVM using the compilation time heat map shown in Fig. 9, which graphically summarizes the observed patterns. The values represent the percentage of total query execution time that is spent on compilation, across various JIT engine

configurations and scale factors that increase by an order of magnitude. As the scale factor increases, the relative compilation overhead decreases, indicating that compiled execution begins to outperform interpreter-based execution for an increasing number of queries (denoted by the ▲ symbol in the figure). In the case of LLVM, its various JIT configurations only begin to justify their compilation overhead starting at SF=1, with the notable exception of query q1 at SF=0.1. Query q1 has the longest execution time in the benchmark, which is reflected in its relatively low compilation time percentage compared to other queries. Furthermore, at SF=1, both lightweight JIT compilers already demonstrate improved execution time over the interpreter across most queries. In contrast, LLVM – even in its most lightweight configuration LLVM_E – does not consistently outperform the interpreter, indicating that its compilation overhead remains significantly higher.

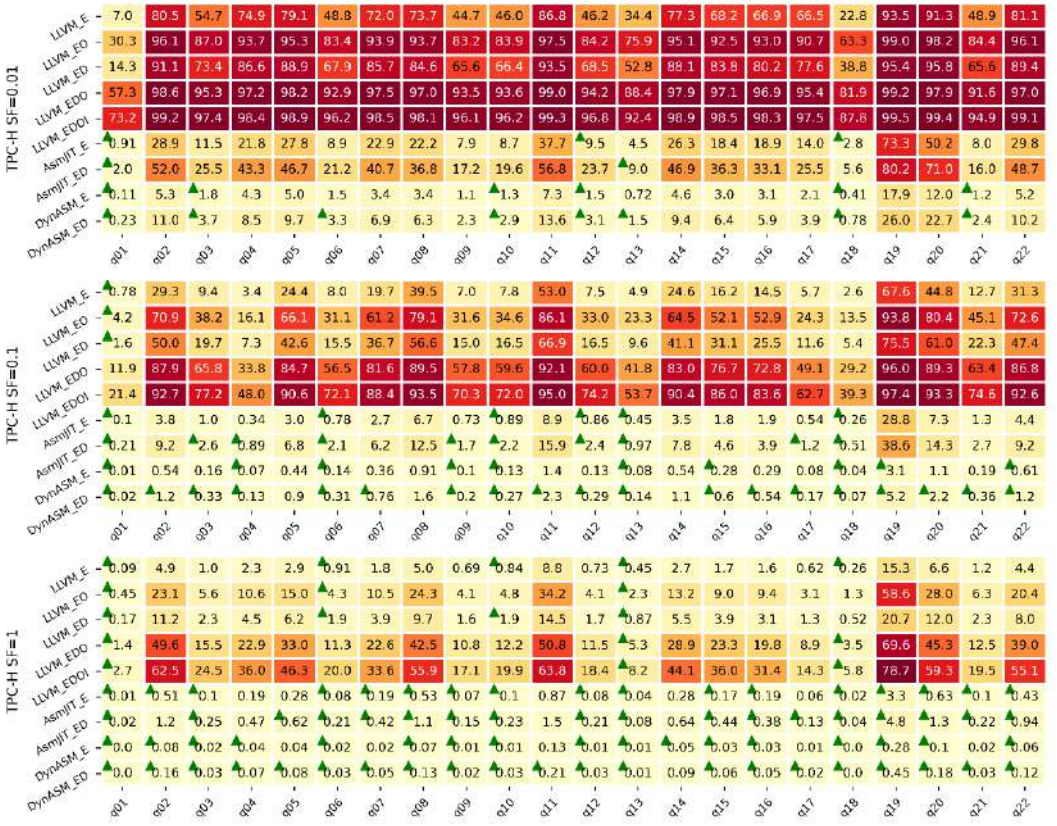
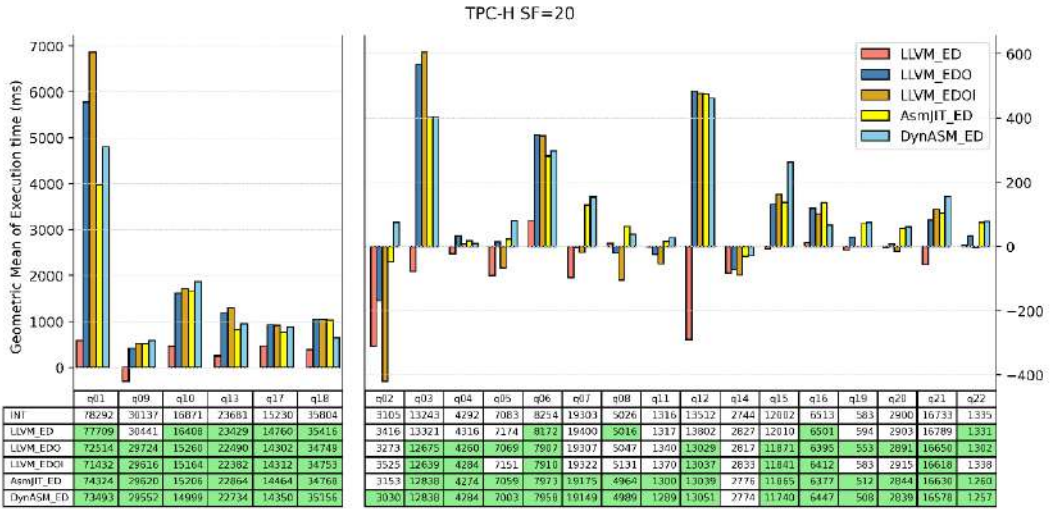


Fig 9. Compilation time as a percentage of total query execution time across TPC-H benchmark scale factors for various JIT engine configurations. Both execution and compilation times represent geometric mean values based on 20 query runs. ▲ indicates that a particular configuration achieves better performance than the interpreter for a given query.

Next, we compare only the most compilation-heavy JIT configurations on the largest of scale factors in our tests, SF=20 (Fig. 10), to examine performance behavior on a relatively large dataset. Interpreter execution time served as the baseline: positive values indicate that JIT execution is faster than interpretation by the corresponding number of milliseconds, while negative indicate it is slower by that amount. The results show that only one query – q14 – does not benefit from any JIT engine. DynASM_ED consistently improves performance across all queries except q14. Similarly, AsmJIT_ED also shows improvements across the board, with an additional exception of q2. LLVM_EDO improves performance on 17 out of 22 queries, achieving the best absolute per-query performance on q4, q6, q12, q17, and q18. Meanwhile, LLVM_EDOI shows the best absolute per-

query performance only on q1, q3, and q13. This suggests that further increasing the scale factor would likely lead to broader performance benefits across all queries when using the LLVM JIT with full optimization enabled, despite its higher compilation overhead.

Interestingly, Q1 – the most time-consuming query – shows a relatively small performance gap between the lightweight JITs and LLVM. For this query LLVM_EDOI incurs a total compilation time of approximately 100ms, whereas DynASM_ED requires less than 0.1 ms. When calculating the relative performance improvements – DynASM_ED ~6.13%, LLVM_EDOI ~8.76%, and idealized "zero-compile-time" LLVM_EDOI ~8.88% – we observe that DynASM_ED achieves approximately two-thirds of the performance gain offered by LLVM on this long-running query. This indicates that, at relatively high scale factors, lightweight JITs can maintain competitive performance compared to LLVM.



	q01	q09	q10	q13	q17	q18
INT	78292	30137	16871	23681	15230	35804
LLVM_ED	77709	30441	16408	23029	14760	35416
LLVM_EDO	72514	29724	15260	22690	14302	34749
LLVM_EDOI	71432	29616	15164	22382	14312	34753
AsmJIT_ED	74324	29620	15206	22864	14464	34760
DynASM_ED	73493	29557	14999	22734	14350	35156

	q02	q03	q04	q05	q06	q07	q08	q11	q12	q14	q15	q16	q19	q20	q21	q22
INT	3105	13243	4292	7083	8254	19303	5026	1316	13512	2744	12002	6513	583	2900	16733	1335
LLVM_ED	3416	13321	4316	7174	8172	19400	5016	1317	13802	2827	12010	6501	594	2903	16789	1331
LLVM_EDO	3273	12675	4260	7069	7907	19307	5047	1300	13029	2817	11871	6395	553	2891	16650	1302
LLVM_EDOI	3525	12639	4284	7151	7910	19322	5131	1370	13037	2833	11841	6412	583	2915	16618	1338
AsmJIT_ED	3153	12838	4274	7059	7973	19175	4964	1300	13039	2776	11865	6377	512	2844	16630	1260
DynASM_ED	3030	12838	4284	7003	7958	19149	4989	1289	13051	2774	11740	6447	508	2838	16578	1257

Fig 10. Execution time difference: (Interpreter - JIT) on TPC-H SF=20. Green table cells indicate that the JIT query execution time is lower than that of the interpreter.

Next, we take a closer look at run-time performance and compilation characteristics of the JIT configurations shown in Figure 11. First, there is a significant gap in compilation time between lightweight JIT engines and LLVM-based configurations. The difference between the basic configurations of DynASM_E and LLVM_E is 69.4x. When optimizations are enabled in LLVM (LLVM_EO), the gap becomes even more pronounced – 397.9x. Second, the difference in compilation time between the lightweight configurations of DynASM and AsmJIT is 6.8x and 7.75x respectively. We attribute this primarily to AsmJIT's automatic register allocation, which introduces additional overhead compared to DynASM. Regarding run-time performance, we observe that all JIT configurations generate more efficient machine code than the interpreter. The best run-time result is achieved by the LLVM_EDOI configuration, completing execution of the entire benchmark in less than 12 seconds, excluding compilation time. Lightweight configurations with deform function generation (*_ED) show performance close to that of the optimized LLVM configuration (LLVM_EDO). However, when both run-time and compilation time are considered together, LLVM-based configurations lose their run-time advantage. For example, the LLVM_EDOI configuration, which has the best run-time performance, reaches a total execution time of 14.49 seconds when compilation time is included. In contrast, lightweight JIT engines, which has almost no compilation overhead, achieve noticeably lower total execution times.

Finally, we compare the overall performance of the JIT engines across all scale factors (Fig. 12) to determine whether there is a threshold at which the lightweight JIT engines begin to lose ground to LLVM as the scale factor increases.

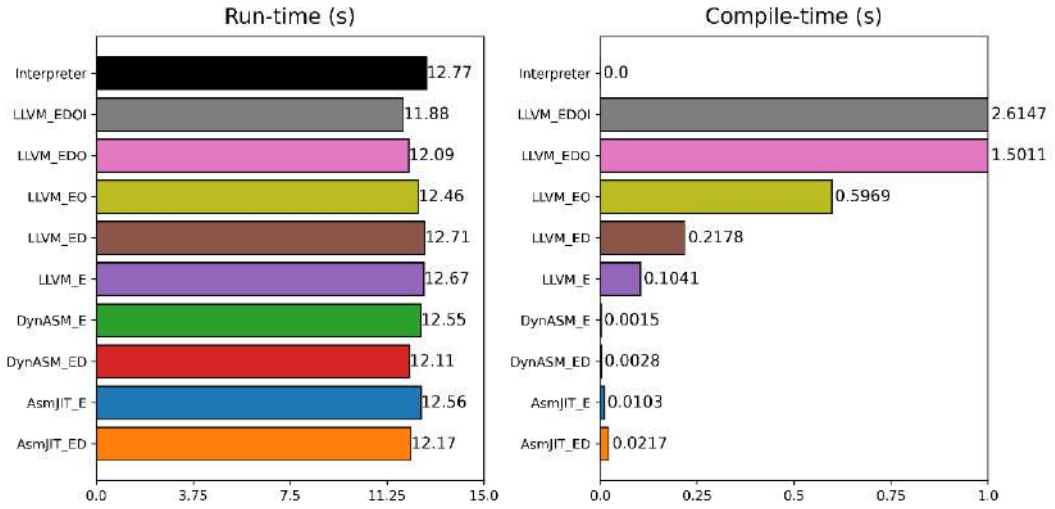


Fig 11. Compile-time and run-time accumulated over all TPC-H queries at scale factor 1. Each query executed 21 times; 1st warm-up run is discarded and geometric mean is computed over the remaining runs.

The results calculated using geometric mean show that when overall system throughput is prioritized over peak per-query performance, such a shift is not clearly evident. Across all scale factors, DynASM_ED maintains the leading position, thanks to its fast code generation and emission speed – even though it generates lower-quality code compared to AsmJIT_ED. AsmJIT secures second place starting from SF=0.5, but ranks third on the two smallest scale factors, where it loses the second position to DynASM due to slower code generation. LLVM_EDO only reaches third place starting at SF=15, while LLVM_EDOI begins to show overall improvement starting at SF=20, suggesting that higher scale factors are necessary to fully benefit from LLVM's optimization pipeline.

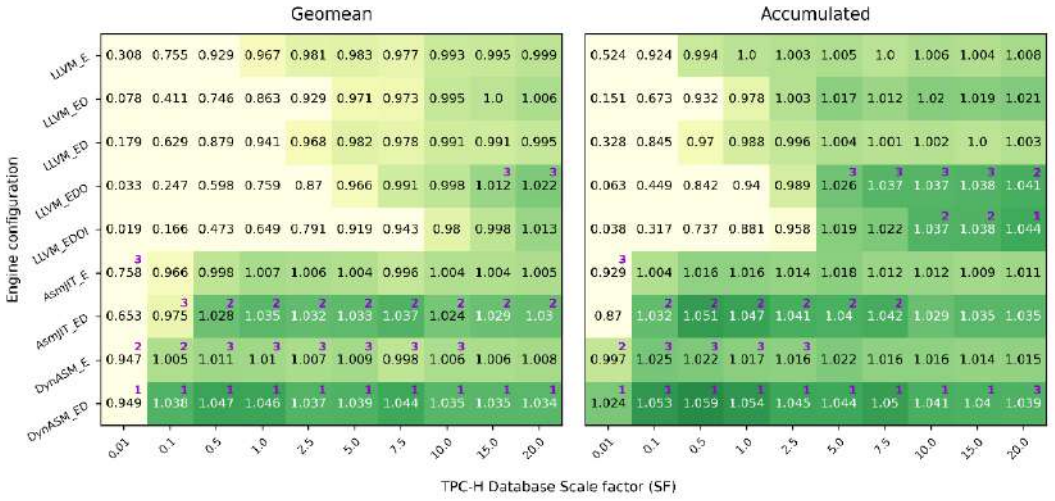


Fig 12. JIT speedups for all queries across all tested scale factors relative to the interpreter, shown in two ways: (left) geometric mean of execution times, and (right) total sum of execution time. The numbers in the top right corner of the cells show the top 3 results ranked by improvement at each scale factor. Cell values indicate the speedup (or slowdown) relative to the interpreter, expressed as × times.

The results, calculated as a sum of execution time of all queries, show that starting from scale factor 10, LLVM ranks second and third, with only DynASM outperforming it. By scale factor 20, LLVM surpasses DynASM and claims the top two positions. This can be explained by the varying execution times of TPC-H queries. They can be roughly grouped into two categories: short-running and long-running queries. The latter start to take significantly longer time to execute as the scale factor increases and benefit more from high-quality machine code, even at lower scale factors. For example, at scale factor 20, query 9 takes around 30 seconds to complete, while query 2 finishes in just 3 seconds – a tenfold difference. Because of this imbalance, the long-running queries make up most of the total benchmark execution time. This gives LLVM a performance advantage over lightweight JIT engines, as improvements in these costly queries have a bigger effect on the total runtime. LLVM's advantage is expected to grow with higher scale factors, since compilation time becomes less significant.

5. Conclusion

In this paper, we evaluated two lightweight JIT compilers – DynASM and AsmJIT – against LLVM, using PostgreSQL as a common experimental environment. Our findings show the AsmJIT-based implementation, which utilizes a higher-level abstraction for automatic register allocation, produces higher-quality code compared to the DynASM-based implementation, which relies on manual register allocation. Nevertheless, DynASM's exceptional code generation speed allows it to outperform AsmJIT across all tested scales.

Both lightweight JIT compilers begin to improve system performance starting from a scale factor (SF) of 0.5, demonstrating their suitability even for very small queries. The query planner's cost model can be used to configure such JIT compilers to operate at low cost thresholds, without being overly sensitive to inevitable cost estimation inaccuracies.

While LLVM demonstrates better peak performance on certain queries, it generally lags behind lightweight JIT engines until reaching the largest tested scale factor (SF=20), where its performance gains become evident. This suggests that even larger scale factors are necessary for LLVM's extensive optimization pipeline to yield meaningful benefits.

As part of our future work, we plan to explore an AsmJIT-based implementation without compiler abstraction, in order to enable a more direct comparison of code generation and emission speed with DynASM. Additionally, we aim to investigate platform-independent JIT assembler alternatives, as our current implementations are limited to x86-64 architecture, and porting and maintaining them on other architecture requires considerable effort. We also intend to evaluate medium-weight JIT engines – often referred to as "mini-LLVM" style compilers – equipped with a minimal set of essential optimizations, again using PostgreSQL as our testbed.

References

[1]. PostgreSQL – open Source DBMS. [Online], Available at: https://www.postgresql.org, accessed 05.05.2025.

[2]. G. Graefe, "Volcano - An Extensible and Parallel Query Evaluation System," IEEE Trans. Knowl. Data Eng., vol. 6, no. 1, pp. 120-135, 1994. DOI: 10.1109/69.273032. [Online], Available at: https://doi.org/10.1109/69.273032.

[3]. T. Neumann, "Efficiently Compiling Efficient Query Plans for Modem Hardware," PVLDB, vol. 4, no. 9, pp. 539-550, 2011. DOI: 10.14778/2002938.2002940. [Online], Available at: http://www.vldb.org/pvldb/vol4/p539-neumann.pdf.

[4]. PostgreSQL mailing lists, "[GSoC] Push-based query executor discussion." [Online], Available at: *https://www.postgresql.org/message-id/87shm1zfnz.fsf%40ispras.ru*, accessed 05.05.2025.

[5]. A. Shaikhha, M. Dashti and C. Koch, «Push vs. pull-based loop fusion in query engines», CoRR, abs/1610.09166, 2016 arXiv: 1610.09166. Available at: *https://arxiv.org/abs/1610.09166*.

[6]. PostgreSQL 10.0 Release Notes, [Online] Available at: *https://www.postgresql.org/docs/release/10.0*, accessed 05.05.2025.

[7]. GitHub, Mirror of the official PostgreSQL GIT repository, "Faster expression evaluation and targetlist projection." Commit SHA: b8d7f053c5c2bf2a7e8734fe3327f6a8bc711755 [Online], Available at: *https://github.com/postgres/postgres/commit/b8d7f053c5c2bf2a7e8734fe3327f6a8bc711755*, accessed 05.05.2025.
[8]. PGCon 2017. A. Freund Integrating Just In Time Compilation, [Online] Available at: *https://www.pgcon.org/2017/schedule/attachments/462_jit-pgcon-2017-05-25.pdf*, accessed 05.05.2025.
[9]. The LLVM Foundation, The LLVM Compiler Infrastructure. [Online] Available at: *https://llvm.org*, accessed 05.05.2025.
[10]. PGCon 2018. A. Freund A.The State of Postgres JIT – 2018 Edition, [Online] Available at: *https://anarazel.de/talks/2018-06-01-pgcon-state-of-jit/state-of-jit.pdf*, accessed 05.05.2025.
[11]. A. Engelke and T Schwarz. 2024. Compile-Time Analysis of Compiler Frameworks for Query Compilation. In Proceedings of the 2024 IEEE/ACM International Symposium on Code Generation and Optimization (CGO '24). IEEE Press, 233–244. DOI: *https://doi.org/10.1109/CGO57630.2024.10444856*.
[12]. A. Kohn, V. Leis, T. Neumann "Adaptive Execution of Compiled Queries", 2018 IEEE 34th International Conference on Data Engineering (ICDE), Paris, France, 2018, pp. 197-208, DOI: 10.1109/ICDE.2018.00027.
[13]. T. Kersten, V. Leis, T. Neumann "Tidy Tuples and Flying Start: fast compilation and fast execution of relational queries in Umbra" The VLDB Journal 30, 5 (Sep 2021), 883–905. [Online], Available at: *https://doi.org/10.1007/s00778-020-00643-4*.
[14]. Transaction Processing Performance Council, "TPC-H Benchmark." [Online]. Available at: *http://www.tpc.org/tpch*, accessed 05.05.2025.
[15]. GCC, the GNU Compiler Collection, [Online], Available at: *https://gcc.gnu.org/*, accessed 05.05.2025.
[16]. V8 JavaScript Engine, [Online], Available at: *https://v8.dev/*, accessed 05.05.2025.
[17]. OpenJDK, The HotSpot Group, [Online], Available at: *https://openjdk.org/groups/hotspot/*, accessed 05.05.2025.
[18]. DotNet, RyuJIT compiler overview, [Online], Available at: *https://github.com/dotnet/runtime/blob/main/docs/design/coreclr/jit/ryujit-overview.md*, accessed 05.05.2025.
[19]. Parrot Virtual Machine, [Online], Available at: *http://www.parrot.org/*, accessed 05.05.2025.
[20]. MoarVM, a VM for NQP and Rakudo, [Online], Available at: *https://www.moarvm.org/*, accessed 05.05.2025.
[21]. GNU LibJIT – backend for DotGNU Portable.NET JIT engine, [Online], Available at: *https://www.gnu.org/software/libjit*, accessed 05.05.2025.
[22]. DynASM – dynamic assembler for code generation engines. [Online], Available at: *https://luajit.org/dynasm.html*, accessed 05.05.2025.
[23]. AsmJIT – lightweight library for low-latency machine code generation. [Online], Available at: https://asmjit.com, accessed 05.05.2025.
[24]. GNU lightning – library that generates assembly language code at run-time. [Online], Available at: https://www.gnu.org/software/lightning/, accessed 05.05.2025.
[25]. sljit – a low-level, platform-independent JIT compiler. [Online], Available at: https://zherczeg.github.io/sljit/, accessed 05.05.2025.
[26]. Xbyak – a JIT assembler for x86/x64 architectures. [Online], Available at: *https://github.com/herumi/xbyak*, accessed 05.05.2025.
[27]. M. Pall, "LuaJIT 2.0 intellectual property disclosure and research opportunities", [Online], Available at: *http://lua-users.org/lists/lua-l/2009-11/msg00089.htm*l, accessed 05.05.2025.
[28]. Erlang/OTP documentation "BeamAsm, the Erlang JIT", [Online] Available at: *https://www.erlang.org/doc/apps/erts/beamasm.html*, accessed 05.05.2025.

Информация об авторах / Information about authors

Михаил Вячеславович ПАНТИЛИМОНОВ – научный сотрудник отдела компиляторных технологий ИСП РАН. Научные интересы: статический анализ, компиляторные технологии, СУБД.

Mikhail Vyacheslavovich PANTILIMONOV – researcher at Compiler Technology department of ISP RAS. Research interests: static analysis, compiler technologies, DBMS.

Рубен Артурович БУЧАЦКИЙ – кандидат технических наук, научный сотрудник отдела компиляторных технологий ИСП РАН. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации.

Ruben Arturovich BUCHATSKIY – Cand. Sci. (Tech.), researcher at Compiler Technology department of ISP RAS. Research interests: static analysis, compiler technologies, optimizations.

Денис Владиславович ЗАВЕДЕЕВ – аспирант Института системного программирования им. В.П. Иванникова Российской академии наук. Сфера научных интересов: компиляторы, языковые виртуальные машины.

Denis Vladislavovich ZAVEDEEV – postgraduate student at Ivannikov Institute for System Programming of the Russian Academy of Sciences. Research interests: compilers, language virtual machines.

DOI: 10.15514/ISPRAS-2025-37(6)-22

Метод обучения персептрона на табличных данных с пропусками

А.И. Перминов, ORCID: 0000-0001-8047-0114 <perminov@ispras.ru>
А.П. Коваленко, ORCID: 0009-0007-8777-8622 <a.p.kovalenko@ispras.ru>
Д.Ю. Турдаков, ORCID: 0000-0001-8745-0984 <turdakov@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН, Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. Обработка пропусков в табличных данных остаётся важной задачей при построении надёжных моделей машинного обучения. В данной работе рассматривается новый подход к заполнению пропущенных значений, основанный на идее унарной классификации. Предложенный метод использует ансамбль персептронов, обучаемых отдельно для каждого класса, для оценки правдоподобия восстанавливаемых значений относительно эмпирического носителя класса. В качестве фона используется равномерное распределение на ограниченной области признакового пространства. Это позволяет интерпретировать выход модели как аппроксимацию апостериорной вероятности принадлежности объекта к классу и использовать её в процессе итеративного заполнения пропусков и обучения классификатора. Теоретически обоснована состоятельность построенной оценки. Проведены эксперименты на синтетических двумерных выборках с пропусками, распределёнными по механизму MCAR. Полученные результаты демонстрируют преимущества предложенного подхода по сравнению с классическими методами заполнения, особенно при высокой доле пропусков и сложной геометрии классов.

Ключевые слова: пропущенные данные; заполнение пропусков; унарная классификация; персептрон; машинное обучение; байесовский классификатор; оценка апостериорной вероятности; MCAR; нейросетевая регрессия.

Для цитирования: Перминов А.И., Коваленко А.П., Турдаков Д.Ю. Метод обучения персептрона на табличных данных с пропусками. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 93–106. DOI: 10.15514/ISPRAS–2025–37(6)–22.

Method for Training Perceptron on Tabular Data with Missing Values

A.I. Perminov, ORCID: 0000--0001-8047-0114 <perminov@ispras.ru>
A.P. Kovalenko, ORCID: 0009-0007-8777-8622 <a.p.kovalenko@ispras.ru>
D.Y. Turdakov, ORCID: 0000-0001-8745-0984 <turdakov@ispras.ru>

Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. Handling missing values in tabular data remains a critical challenge for building robust machine learning models. This paper presents a novel approach to imputation based on unary classification. The proposed method employs an ensemble of perceptrons trained independently for each class to estimate the likelihood of reconstructed values with respect to the empirical support of that class. A uniform distribution over a bounded region of the feature space is used as a background model, enabling the interpretation of the model's output as an approximation of the posterior probability that an object belongs to a given class. This probabilistic interpretation is then leveraged within an iterative procedure for missing value imputation and classifier training. The theoretical validity of the proposed estimator is rigorously justified. Experiments on synthetic two-dimensional datasets with missing values generated under the MCAR (Missing Completely At Random) mechanism demonstrate the superiority of the proposed method over classical imputation techniques, particularly in scenarios with high missingness rates and complex class boundaries.

Keywords: missing data imputation; unary classification; perceptron; machine learning; Bayesian classifier; posterior probability estimation; MCAR; neural network regression.

For citation: Perminov A.I., Kovalenko A.P., Turdakov D.Y. Method for training perceptron on tabular data with missing values. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 93-106 (in Russian). DOI: 10.15514/ISPRAS-2025-37(6)-22.

1. Введение

Наличие пропусков в табличных данных остаётся одной из ключевых проблем при построении прикладных моделей машинного обучения. Отсутствие значений может возникать по множеству причин – от сбоев в сборе данных до неполных анкет или отказов пользователей. Игнорирование пропусков приводит к потере данных и смещению оценок, а применение стандартных подходов к заполнению зачастую не учитывает геометрию и структуру распределения.

На практике широкое распространение получили простые эвристики, такие как заполнение средним значением, модой или ближайшими соседями. Однако при высокой доле пропусков, особенно в условиях сложной формы распределения данных, такие методы могут искажать структуру выборки и снижать качество последующей модели. Эта проблема становится особенно заметной в задачах, где классы имеют запутанную или нелинейную геометрию, например, в синтетических задачах типа "двойной спирали", колец или полуколец.

В данной работе предлагается новый метод работы с пропусками, основанный на унарной классификации. Идея заключается в том, чтобы для каждого класса обучать отдельный персептрон, способный отличать реальные объекты от искусственного "фона", порождённого равномерным распределением на компакте. Такой подход позволяет интерпретировать выход модели как оценку вероятности принадлежности объекта к классу. Это даёт возможность использовать персептрон для оценки того, насколько сгенерированное значение "правдоподобно" для конкретного класса, и тем самым достовернее заполнять пропуски.

Целью данной работы является описание предложенного метода, его теоретическое обоснование, а также экспериментальное сравнение с классическими подходами на синтетических двумерных данных с пропусками, распределёнными по механизму MCAR. Особое внимание уделяется поведению метода при высокой доле пропусков и сложной структуре распределения данных.

Статья организована следующим образом. В разделе 2 формализуется задача заполнения пропусков и вводится используемая терминология. В разделе 3 описывается метод унарной классификации и его адаптация к задаче обучения с неполными данными. В разделе 4 приводится подробное описание предлагаемого алгоритма. Раздел 5 посвящён экспериментальному исследованию и анализу результатов. В заключении обсуждаются ограничения метода и возможные направления дальнейшей работы.

2. Постановка задачи

Рассматривается задача обучения классификатора по неполным данным. Пусть задано множество объектов $D = \{X, Y\} = \{(x_i, y_i)\}_{i=1}^{n},\ x_i \in R^d,\ y_i \in \{1, \dots C\}$, в котором векторы признаков x_i могут содержать пропущенные значения. Предполагается, что пропуски распределены по механизму MAR (missing at random), то есть вероятность пропуска не зависит ни от значений признаков, ни от целевой переменной.

Обозначим через M матрицу бинарных переменных, имеющую размерность входной матрицы данных, в которой $m_{ij} = 1$, если признак j в наблюдении x_i отсутствует, и $m_{ij} = 0$ в противном случае. Пусть X_0 – присутствующие части матрицы входных признаков, соответствующие элементам $m_{ij} = 0$, а X_1 – отсутствующие части, соответствующие элементам $m_{ij} = 1$. Полагаем, что выходной вектор Y известен.

Целью является построение классификатора $f: R^d \to \{1, \dots, C\}$, обладающего высокой точностью на полной версии выборки, несмотря на наличие пропусков в обучающих данных. Для решения данной задачи предлагается итеративный подход, включающий два ключевых этапа:

- Заполнение пропусков в X_1 с использованием модели, оценивающей вероятность принадлежности сгенерированных значений к каждому из классов;
- Обучение классификатора на полученной заполненной выборке.

В отличие от традиционных методов, где пропуски заполняются один раз (например, средними значениями, модой или ближайшими соседями), предлагаемый подход предполагает многократное чередование этапов генерации и обучения. После каждой эпохи обучения параметры модели обновляются, и заполнение производится заново – с учётом обновлённой вероятностной оценки правдоподобности заполненных значений. Этот итеративный процесс продолжается до сходимости или достижения заданного количества шагов.

Заполнение на каждом шаге осуществляется на основе модели унарной классификации, которая обучается отличать носитель распределения реальных наблюдений от сгенерированных из равномерного распределения. Концепция унарной классификации и её адаптация к задаче восстановления пропусков подробно рассматриваются в следующем разделе.

3. Метод унарной классификации (случай одного класса)

Метод обучения классификатора при наличии пропущенных входных данных обучающей выборки основан на построении байесовского унарного классификатора.

В работе [1] предложен метод экстраполяции байесовского бинарного классификатора, когда к данным, относящимся к двум разным классам (с метками "+1" и "-1"), добавляется третий, искусственно созданный «фоновый» класс с меткой "0", представляющий собой случайную выборку из заданного на компакте равномерного распределения. Модифицированный таким образом байесовский классификатор, помимо решений об отнесении наблюдений по значению дискриминантной функции к первому или второму классу, может принимать решение об отказе от классификации при близости значений дискриминантной функции к

нулю. Доказано, что на носителе распределения модифицированный байесовский классификатор эквивалентен байесовскому классификатору, за пределами носителя принимается решение об отказе от классификации. Поэтому входным наблюдениям, принадлежащим компакту, но лежащим за пределами эмпирический границы носителя исходного распределения (например, выбросам), будет «отказано» в классификации.

3.1 Унарный байесовский классификатор

Унарный байесовский классификатор отличается от рассмотренного выше модифицированного бинарного байесовского классификатора тем, что во входной выборке присутствуют наблюдения только одного класса, которые обозначены меткой «1», а наблюдения из второго, «фонового» класса обозначены меткой «0». Формально задача унарной классификации может быть представлена следующим образом.

Предположим, что входные данные представляют собой наблюдения n независимых одинаково распределенных случайных d-мерных векторов из R^d, имеющих равномерно непрерывную плотность распределения $f(x)$. Предположим также, что носитель распределения $f(x)$ неизвестен, но расположен внутри компакта $K = [0,1]^d$.

Пусть задана случайная величина (X, Y), где X – d-мерный случайный вектор с равномерно непрерывной плотностью смеси распределений $\alpha f(x) + (1-\alpha)p(x)$, где $f(x)$ – плотность распределения наблюдений исходной выборки, $p(x)$ – плотность равномерного распределения наблюдений «фона», заданная на компакте K, α – весовой коэффициент, $0 \leq \alpha \leq 1$, метка Y принимает значение 1 или 0 в зависимости от того, какому классу принадлежит вектор X, то есть выборке или «фону».

Пусть

$$g(x) = P(Y = 1 \vee X = x) = E(Y \vee X = x) = \frac{f(X)}{f(X) + \frac{1-\alpha}{\alpha} \cdot p(X)} \quad (1)$$

есть апостериорная вероятность выборочного класса (функция регрессии Y на X). Если функция $g(x)$ известна, то задача унарной классификации может быть решена следующим образом: если в точке x функция $g(x) > 0$, то эта точка принадлежит носителю распределения, то есть выборочному классу, в противном случае – это «выброс», относительно которого решение принимается некоторой дополнительной процедурой. Однако, функция $g(x)$, как правило, неизвестна и требуется построить ее аппроксимацию по имеющейся выборке данных.

3.2 Построение аппроксимации

Пусть $c(x)$ – непрерывная функция, заданная на компакте K.

Рассмотрим задачу среднеквадратической аппроксимации (минимизация осуществляется по всем $c(X)$):

$$c^*(x) = argmin\, E(c(x) - Y)^2 \quad (2)$$

Поскольку $E(c(x) - Y)^2 = E(c(x) - g(x) + g(x) - Y)^2 = E\big(c(x) - g(x)\big)^2 + E(g(x) - Y)^2$, а второе слагаемое от $c(x)$ не зависит, задача (2) эквивалентна аппроксимации равномерно непрерывной функции регрессии Y на X:

$$c^*(x) = argmin\, E\big(c(x) - g(x)\big)^2 \quad (3)$$

3.3 Аппроксимация с помощью персептрона

В качестве функции $c(x) = c(x; k, L)$ рассмотрим многослойный персептрон (полносвязную нейросеть) с кусочно-линейной функцией активации $|\cdot|$, состоящий из L скрытых слоев по

k нейронов в каждом. По основной аппроксимационной теореме [2] для любого заданного $\epsilon > 0$ существуют такие значения параметров персептрона k и L, что для любого $x \in K$ выполняется условие (4):

$$\sup_{x \in K} |c(x) - g(x)| < \epsilon \tag{4}$$

то есть теоретически ϵ-приближенное решение задачи (3) существует.

Рассмотрим выборочную постановку задачи унарной классификации. Пусть задана выборка $\{(X_1, Y_1), (X_2, Y_2), \cdots, (X_n, Y_n)\}$, где $X_i \in K, Y_i = 1$, которую будем интерпретировать как размеченный набор n наблюдений случайных векторов, распределенных с равномерно непрерывной плотностью $f(x)$. Для формирования выборки из смеси выборочного класса и «фона» с плотностью $\alpha f(x) + (1-\alpha)p(x)$ добавим к этой выборке искусственно сгенерированные данные $\{(X_{n+1}, Y_{n+1}), (X_{n+2}, Y_{n+2}), \cdots, (X_{n+m}, Y_{n+m})\}$, где $m = n \cdot \frac{1-\alpha}{\alpha}$, векторы $X_{n+i}, i = 1,2, \cdots m$, есть наблюдения независимых равномерно распределённых на компакте K случайных векторов, $Y_{n+i} = 0$.

Пусть $C(k, L)$ – множество всех многослойных персептронов $c(x)$ с кусочно-линейной функцией активации $|\cdot|$ в скрытых слоях и числом L и размером k скрытых слоев.

Применяя некоторый алгоритм оптимизации, построим выборочную оценку решения задачи (2):

$$\sum_{i=1}^{n+M} (c_n(X_i) - Y_i)^2 \to min \tag{5}$$

где минимизация функционала осуществляется по всем $c_n(X) \in C(l, L)$, а параметры k и L выбраны оптимально с учетом ограничений, связанных с переобучением.

Пусть функция $c_n^*(X)$ есть решение оптимизационной задачи (5), которую будем называть **функцией нейросетевой регрессии**. Соответствующий этому решению персептрон строит иерархическое разбиение компакта K на N непересекающихся ячеек $K = \{K_1, K_2, \cdots, K_N\}$ [3] (рис. 1).

Рис. 1. Пример разбиения некоторым MLP с $L = 2$, $k = 6$.
Fig. 1. Some MLP partition example, $L = 2$, $k = 6$.

3.4 Состоятельность метода унарной классификации

Для обоснования состоятельности решения $c_n^*(X)$ рассмотрим кусочно-постоянную (в общем случае разрывную) функцию гистограммной регрессии $h_n(X)$, и решим оптимизационную задачу:

$$\sum_{i=1}^{n+M} (h_n(X_i) - Y_i)^2 \to min \tag{6}$$

где минимизация осуществляется по всем кусочно-постоянным функциям, принимающим постоянные значения в ячейках разбиения компакта $K = \{K_1, K_2, \cdots, K_N\}$.

Пусть $X \in K_r$. Тогда задачу (6) для этой ячейки можно представить в виде:

$$n_1(X) \cdot (h_n(X) - 1)^2 + n_0(X) \cdot (h_n(X) - 0)^2 \to min, \tag{7}$$

где $n_1(X) = \sum_{i=1}^{n+m} I_{X_i \in K_r, Y_i = 1}$, $n_0(X) = \sum_{i=1}^{n+m} I_{X_i \in K_r, Y_i = 0}$.

После дифференцирования функции (7) по $h_n(X)$ получаем решение задачи (6):

$$h_n^*(X) = \frac{n_1(X)}{n_1(X) + n_0(X)} = \frac{f_n(X)}{f_n(X) + \frac{1-\alpha}{\alpha} \cdot p_n(X)}, \tag{8}$$

где $f_n(X) = \frac{n_1(X)}{n \cdot \mu(K_r)}$ – адаптивная гистограммная оценка плотности $f(x)$ в ячейке K_r, $p_n(X) = \frac{n_0(X)}{n \cdot \mu(K_r)}$ – адаптивная гистограммная оценка равномерной плотности в ячейке K_r, $\mu(K_r)$ – мера ячейки K_r. Пример вычисления функции гистограммной регрессии показан на рис. 2.

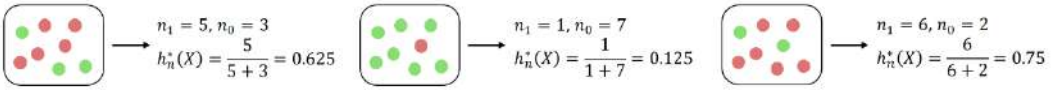


Рис. 2. Пример вычисления $h_n^*(X)$ в некоторой ячейке K_r.
Fig. 2. $h_n^*(X)$ evaluation example at some cell K_r.

В работах [4-5] сформулированы асимптотические условия строгой состоятельности адаптивных гистограммных оценок плотности распределения, при выполнении которых для любого произвольно малого $\epsilon > 0$ и $X \in K$ с вероятностью 1 имеют место соотношения:

$$\begin{aligned} |f(X) - f_n(X)| < \epsilon \\ |p(X) - p_n(X)| < \epsilon \end{aligned} \tag{9}$$

Отсюда следует, что при достаточно больших n для любого $X \in K$ с вероятностью 1:

$$|g(X) - h_n^*(X)| < \epsilon_1, \tag{10}$$

где $\epsilon_1 = \frac{4\epsilon}{\mu(K)}$, $\mu(K)$ – мера компакта K.

Для состоятельности, в частности, требуется, чтобы диаметр ячеек убывал с ростом n, но при этом число точек внутри ячеек стремилось к бесконечности. При размерности пространства $d = 10$ эти требования выполняются уже при малых значениях k и L. Например, для нормальной плотности распределения, $d = 10$, $k = 10$ и $L = 2$ количество ячеек N превышает десятки тысяч, а для их 90%-го заполнения (чтобы в ячейку попала хотя бы одна фоновая точка) требуются миллионы фоновых точек. Следует отметить, что в «заполненных» ячейках, в которых представлены как выборочные, так и фоновые точки, значения функций $h_n^*(X)$ и $c_n^*(X)$ близки (рис. 3). В ячейках, в которые попали только фоновые точки, гистограммная регрессия $h_n^*(X)$, а значение нейросетевой регрессии $c_n^*(X)$ определяется интерполяцией значений в соседних ячейках благодаря непрерывности функции. В областях высокой плотности $f(X)$ значения $c_n^*(X)$ существенно больше нуля, в областях низкой плотности нейросетевая регрессия близка к нулю.

Поэтому при достаточно больших n, правильно подобранных значениях параметров сети k и L и, учитывая соотношения (3) - (10), нейросетевую регрессию $c_n^*(X)$ можно считать состоятельной оценкой апостериорной вероятности $g(X)$, а значит, при решении прикладных задач есть основания полагать, что имеет место соотношение:

$$c_n^*(X) \approx h_n^*(X) \approx g(X) = P(Y = 1 \vee X), \tag{11}$$

то есть значение нейросетевой регрессии можно рассматривать как оценку апостериорной вероятности выборочного класса.

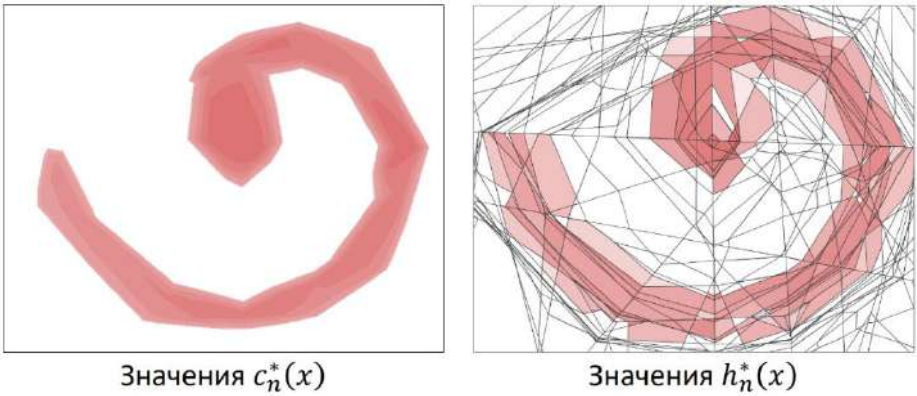


Рис. 3. Сравнение $c_n^*(X)$ и $h_n^*(X)$.
Fig. 3. Comparison of $c_n^*(X)$ and $h_n^*(X)$.

4. Метод обучения MLP при наличии пропусков в обучающей выборке.

Представим входные данные в виде объединения выборок, соответствующим отдельным классам (то есть в каждой такой выборке метки наблюдений совпадают). Пусть число классов равно C. Каждую выборку, в свою очередь, разделим на две подвыборки: первая состоит из наблюдений без пропусков признаков (комплектная подвыборка класса), а вторая – из наблюдений с пропусками (некомплектная подвыборка класса). Обучение MLP осуществляется в соответствии с предположением MAR независимо в каждом классе.

Предлагаемый метод обучения MLP при наличии пропусков в обучающей выборке применяется последовательно к каждому из C классов и состоит из трёх шагов (схематичная иллюстрация к методике дана на рис. 4).

1) **Начальное обучение.** Для комплектной подвыборки $\{X_i, i = 1,2,\cdots,n\}$ j-го класса, $j \in \{1,2,\cdots,C\}$, решить задачу унарной классификации и построить MLP_j, реализующий кусочно-линейную непрерывную функцию $c_n^j(X)$, заданную на компакте K.
2) **Дообучение.** Дообучение осуществляется по всей обучающей выборке j-го класса отдельными эпохами. Перед текущей эпохой выполнить временное (для данной эпохи) заполнение некомплектных наблюдений. Для каждого некомплектного наблюдения X:
   a. Разделить множество индексов координат вектора $X = (x_1, x_2, \cdots, x_d)$ на два подмножества M_0 и M_1, включающие соответственно индексы заполненных и пропущенных координат.
   b. Заполнить координаты X из M_1 наблюдениями равномерно распределенной случайной величины на отрезке $[0,1]$, в результате чего будет получен комплектный вектор X'. Вычислить $c_n^j(X')$. Сгенерировать наблюдение биномиальной случайной величины с вероятностью успеха $p = c_n^j(X')$.
   c. При успешном исходе временно заменить в обучающей выборке некомплектный вектор X на комплектный вектор X' и перейти к рассмотрению следующего некомплектного наблюдения. В противном случае повторить шаг 2.b.
   d. Выполнить дообучение сети по «доукомплектованной» обучающей выборке.

3) Перейти к следующей эпохе дообучения, повторяя шаги a-d, до полного завершения обучения MLP_j для j-го класса с функцией нейросетевой регрессии $c_n^j(X)$.

Повторяя шаги 1-3 для всех классов, получим C обученных нейросетей MLP_j и соответствующих им непрерывных кусочно-линейных функций $\{c_n^1(X), c_n^2(X), \cdots, c_n^C(X)\}$, каждая из которых есть выборочная оценка апостериорной вероятности соответствующего класса в точке X.

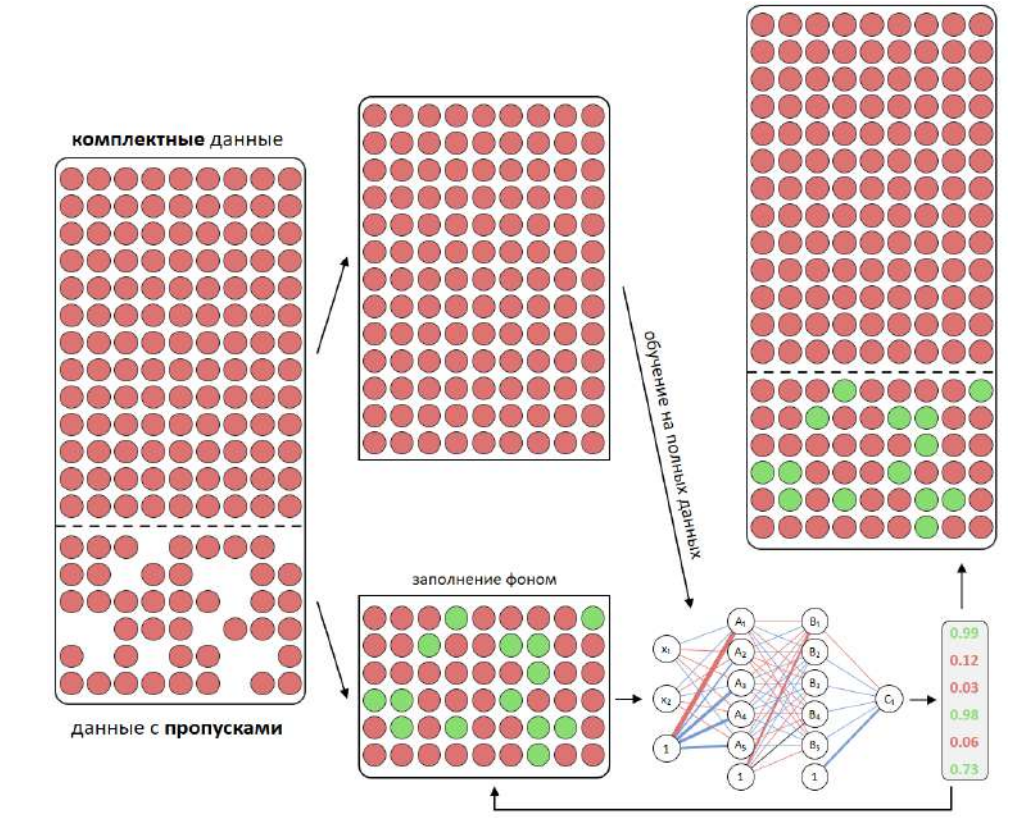


Рис. 4. Схема обучения $c_n^j(X)$ по неполным данным.
Fig. 4. $c_n^j(X)$ training on missed data diagram.

Для решения задачи классификации комплектного наблюдения X возможны различные стратегии. Простейшая состоит в выборе класса, для которого апостериорная вероятность максимальна. Другой вариант – выбрать в качестве решения все классы, значения апостериорной вероятности для которых больше некоторого заданного порога, и продолжить решение задачи классификации, например, в другом признаковом пространстве. Возможно, целесообразно учитывать априорные вероятности классов, различные функции стоимости ошибок для разных классов и т.п. Рассмотрение этих вопросов выходит за рамки данного исследования и для простоты используется выбор класса с максимальным значением $c_n^j(X)$.

Реализация описанного метода обучения и сценарии всех экспериментов доступны в открытом репозитории [6], включающим код генерации синтетических данных, обучение MLP с применением различных стратегий заполнения, а также визуализацию апостериорных распределений.

Таким образом, предложенный метод позволяет отказаться от прямого восстановления недостающих признаков, заменяя его вероятностной процедурой включения неполных

наблюдений в процессе обучения. Это особенно важно в случаях, когда форма распределения классов не допускает корректного заполнения с помощью глобальных статистик.

5. Эксперименты

Цель данного раздела – эмпирически оценить эффективность предлагаемого метода на синтетических наборах данных с различной топологической и геометрической структурой при различной доле пропусков. Особое внимание уделяется случаям, когда стандартные методы заполнения демонстрируют снижение качества из-за неспособности учесть сложную форму распределения классов.

5.1 Наборы данных

В экспериментах использовались следующие двумерные синтетические наборы данных:

- **Гауссианы** – два нормально распределённых кластера с равной дисперсией и небольшим перекрытием (рис. 5, слева);
- **Спирали** – классы формируют витки спиралей с общей точкой начала координат, разделение классов сильно нелинейное (рис. 5, по центру);
- **Кольцо и круг** – один класс расположен внутри круга, второй образует кольцо с зазором между границами (рис. 5, справа).

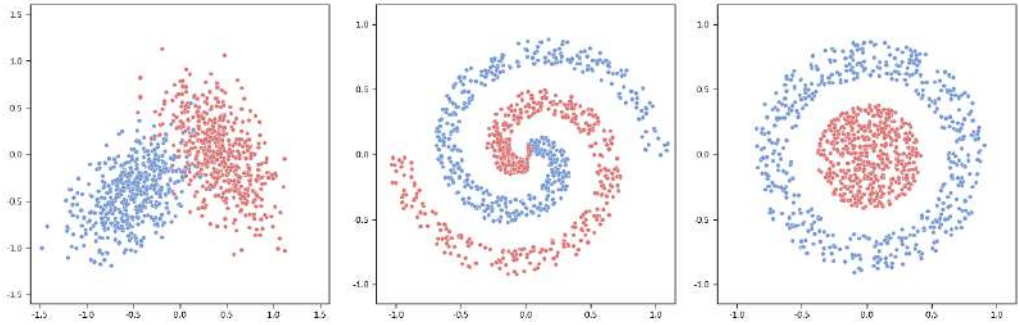


Рис. 5. Наборы данных: гауссианы (слева), спираль (центр), кольцо и круг (справа).
Fig. 5. Datasets: gaussians (left), spiral (center), ring and circle (right).

Каждый набор содержал 1000 наблюдений для обучающей части и 5000 наблюдений для тестовой.

5.2 Обработка пропущенных значений

Во всех наборах данных искусственно вводились пропуски в признаках с уровнями 20%, 40%, 50%, 60%, 80% и 90%. Пропуски вносились случайно и только в признаках (целевые метки всегда сохранялись). Были рассмотрены следующие методы обработки пропусков:

- $mean$ – заполнение по среднему значению признака.
- $mode$ – заполнение наиболее частым значением.
- kNN $(k = 3)$ – заполнение по ближайшим трём соседям в евклидовом пространстве.
- kNN $(k = 7)$ – аналогично, но с $k = 7$.
- $reproduction$ (предлагаемый метод) – метод, основанный на унарной классификации из раздела 4.

5.3 Сценарии обучения

Для каждой комбинации набора данных и уровня пропусков модель обучалась в следующих режимах:

- $full$ – обучение на полном наборе без пропусков.
- $complete$ – обучение только на тех примерах, где отсутствуют пропуски.
- $imputed$ – обучение на наборе, где пропуски заполнялись одним из методов.

В качестве модели использовался полносвязный персептрон с $L = 2$ скрытыми слоями по $k = 20$ нейронов в каждом и одним выходным слоем. Обучение осуществлялось на протяжении 500 эпох. Для метода репродукции персептрон обучался в течение 50 эпох на данных без пропусков, а затем каждую эпоху запускался процесс вероятностного заполнения пропусков и обучение продолжалось уже на обновлённых заполненных данных.

5.4 Оценка качества

Каждая комбинация набора данных, уровня пропусков и метода заполнения запускалась 50 раз с различными начальными инициализациями весовых коэффициентов. В качестве основной метрики использовалась правильность классификации (accuracy) на тестовом множестве из соответствующего набора данных из 5000 элементов. Все тестовые наборы содержали только полные данные.

5.5 Результаты

Результаты со значениями accuracy (среднее ± стандартное отклонение) по 50 запускам представлены в табл. 1, табл. 2 и табл. 3. Визуальный анализ показывает, что предлагаемый метод репродукции демонстрирует более высокую устойчивость при высоких уровнях пропусков, особенно на сложных наборах данных, как например "кольцо и круг". Традиционные методы заполнения (среднее, мода) показывают ожидаемое снижение качества, особенно при пропусках выше 60%. Метод kNN даёт умеренное улучшение, но чувствителен к плотности выборки.

Табл. 1. Результаты на наборе данных «Гауссианы».
Table 1. Results on the "Gaussians" dataset.

пропуски	full	complete	reproduce	mean	most frequent	knn 3	knn 7
20%	0.925 ± 0.003	0.919±0.006	**0.925±0.004**	0.917±0.007	0.931±0.007	0.919±0.004	0.921±0.006
40%		**0.919±0.006**	0.917±0.009	0.897±0.008	0.913±0.020	0.909±0.010	0.915±0.005
50%		0.917±0.005	**0.918±0.008**	0.904±0.014	0.909±0.018	0.906±0.007	0.917±0.005
60%		0.910±0.013	**0.921±0.009**	0.866±0.018	0.869±0.047	0.888±0.011	0.894±0.012
80%		0.875±0.011	**0.912±0.008**	0.752±0.047	0.781±0.057	0.852±0.015	0.851±0.011
90%		0.842±0.023	**0.906±0.007**	0.593±0.092	0.627±0.128	0.723±0.030	0.806±0.015

Табл. 2. Результаты на наборе данных «Спираль».
Table 2. Results on the "Spiral" dataset.

пропуски	full	complete	reproduce	mean	most frequent	knn 3	knn 7
20%	0.941 ± 0.024	0.936±0.031	**0.945±0.025**	0.922±0.032	0.934±0.029	0.927±0.034	0.941±0.020
40%		0.924±0.023	**0.930±0.021**	0.899±0.034	0.892±0.057	0.880±0.057	0.918±0.025
50%		**0.926±0.016**	0.910±0.034	0.893±0.031	0.873±0.047	0.867±0.032	0.868±0.062
60%		**0.913±0.030**	0.898±0.047	0.890±0.027	0.841±0.081	0.823±0.044	0.853±0.040
80%		**0.869±0.039**	0.861±0.044	0.750±0.095	0.632±0.127	0.727±0.041	0.773±0.062
90%		**0.827±0.042**	0.812±0.067	0.545±0.082	0.373±0.129	0.648±0.049	0.695±0.040

Табл. 3. Результаты на наборе данных «Кольцо и круг».
Table 3. Results on the "Ring and circle" dataset.

пропуски	full	complete	reproduce	mean	most frequent	knn 3	knn 7
20%	0.981 ± 0.014	0.987±0.009	**0.989±0.006**	0.941±0.058	0.983±0.008	0.954±0.042	0.927±0.086
40%		0.984±0.011	**0.986±0.011**	0.865±0.103	0.970±0.015	0.887±0.080	0.846±0.123
50%		0.971±0.019	**0.984±0.016**	0.852±0.094	0.958±0.025	0.868±0.119	0.751±0.149
60%		0.974±0.018	**0.981±0.016**	0.763±0.083	0.907±0.068	0.797±0.059	0.705±0.103
80%		0.892±0.136	**0.964±0.031**	0.609±0.148	0.673±0.177	0.781±0.058	0.618±0.069
90%		0.852±0.080	**0.952±0.038**	0.295±0.126	0.433±0.170	0.522±0.039	0.562±0.055

Для дополнительного визуального анализа на рис. 6, рис. 7 и рис. 8 показаны некоторые (лучшие) модели, полученные после обучения с помощью метода репродукции.

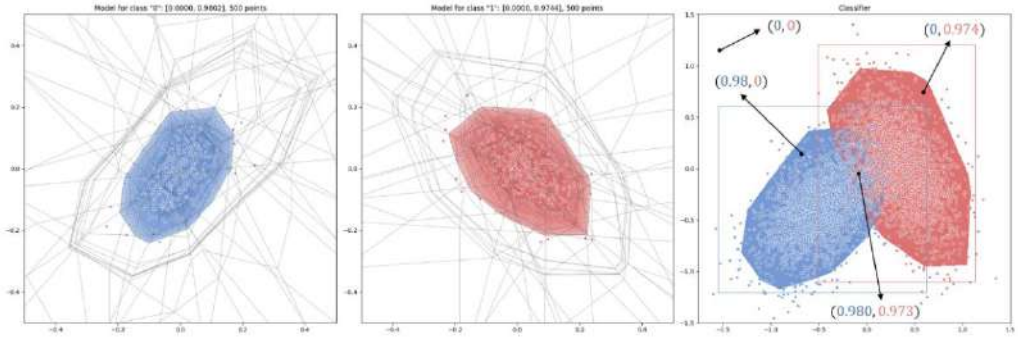


Рис. 6. Модель, полученная при классификации набора данных "гауссианы".
Fig. 6. Model obtained by classifying the gaussian dataset.

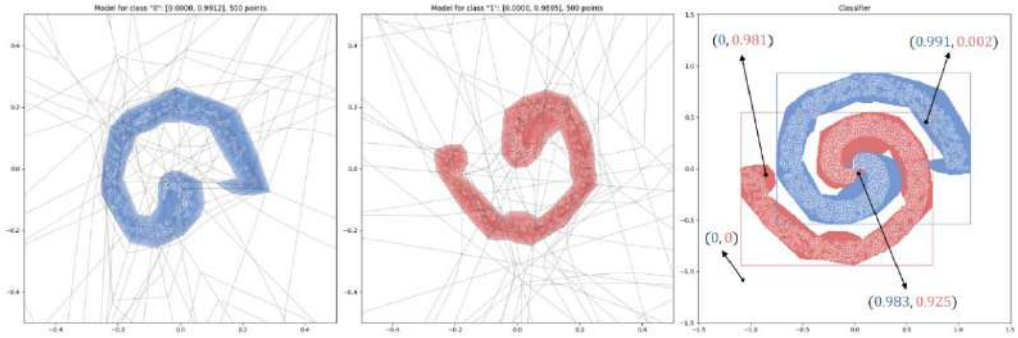


Рис. 7. Модель, полученная при классификации набора данных "спираль".
Fig. 7. Model obtained by classifying the spiral dataset.

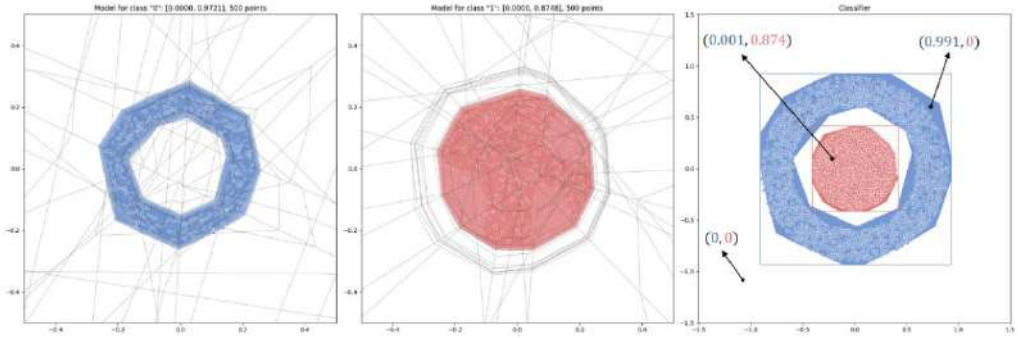


Рис. 8. Модель, полученная при классификации набора данных "кольцо и круг".
Fig. 8. Model obtained by classifying the ring and circle dataset.

Таким образом, предложенный подход воспроизведения недостающих признаков на основе унарной классификации демонстрирует высокую устойчивость к пропускам и способность к адаптации к сложной геометрии данных. Особенно заметно его преимущество на структурах, где границы классов являются нелинейными и неоднородными, как, например, в задачах «кольцо и круг» или «спирали». Метод воспроизведения не требует оценки плотности или предположений о форме распределения и теоретически может быть расширен на более высокие размерности. В сочетании с доступной реализацией и возможностью встраивания в существующие модели, он представляет собой практический и теоретически обоснованный инструмент для работы с неполными данными.

6. Заключение

В данной работе был предложен метод обработки пропусков в табличных данных, основанный на идее унарной классификации и вероятностного восстановления значений на фоне равномерного распределения. Подход ориентирован на работу с пропущенными признаками при обучении многослойного перцептрона (MLP) и направлен на корректное представление неопределённости, связанной с отсутствием информации.

Эксперименты на синтетических двумерных задачах показали, что метод воспроизведения обеспечивает более высокое качество классификации по сравнению с классическими подходами, такими как удаление объектов с пропусками, заполнение средним и kNN-интерполяция. Особенно заметным это преимущество становится в задачах с топологически сложными границами классов, где искажение геометрии данных приводит к существенной деградации качества у традиционных методов.

Предложенный метод не требует предварительного заполнения и интегрируется непосредственно в процесс обучения. Он сохраняет гибкость нейросетевых моделей, одновременно учитывая вероятностную природу отсутствующих значений.

Тем не менее, работа имеет ряд ограничений. В настоящей формулировке метод реализован для пропусков в одном признаке и на фиксированной архитектуре MLP. Кроме того, эффективность метода не проверялась на реальных табличных наборах данных с пропущенными данными, где могут действовать более сложные механизмы отсутствия информации (MAR, MNAR).

Будущая работа может быть направлена на:

- расширение метода на произвольное количество пропущенных признаков;
- адаптацию к другим типам моделей (например, градиентный бустинг, трансформеры);
- применение к реальным задачам, включая медицинские и социологические данные;
- исследование устойчивости при различных механизмах пропусков.

Таким образом, предложенный подход открывает перспективное направление для построения устойчивых моделей, способных эффективно обучаться в условиях неполных данных.

Список литературы / References

[1]. Lukyanov K. S. et al. Extrapolation of the Bayesian classifier with an unknown support of the two-class mixture distribution //Russian Mathematical Surveys. 2024. Vol. 79, No. 6, pp. 991-1015.

[2]. Cybenko G. Approximation by superpositions of a sigmoidal function //Mathematics of control, signals and systems. 1989. Vol. 2, No. 4, pp. 303-314.

[3]. Kovalenko A. Geometric interpretation of a multilayer perceptron with piecewise linear activation functions // 31st scientific and technical conference "MiTSOBIT". Saint Petersburg, 2022. pp. 34—35.

[4]. Devroye L. Nonparametric density estimation //The L_1 View. 1985.

[5]. Devroye L., Györfi L., Lugosi G. A probabilistic theory of pattern recognition. – Springer Science & Business Media, 2013. Vol. 31.
[6]. MissingDataPerceptron, https://github.com/dronperminov/MissingDataPerceptron, last accessed: 01 July 2025.

Информация об авторах / Information about authors

Андрей Игоревич ПЕРМИНОВ является аспирантом института системного программирования РАН. Научные интересы: нейросетевая обработка данных, цифровая обработка изображений, методы доверенного искусственного интеллекта.

Andrey Igorevich PERMINOV – a postgraduate student at the Institute of System Programming of the RAS. Research interests: neural network data processing, digital image processing, trusted artificial intelligence.

Андрей Петрович КОВАЛЕНКО – доктор технических наук, исследователь центра доверенного искусственного интеллекта в институте системного программирования РАН. Научные интересы: методы доверенного искусственного интеллекта.

Andrey Petrovich KOVALENKO – Dr. Sci. (Tech.), a researcher at the Center for Trusted Artificial Intelligence at the Institute of System Programming of the RAS. Research interests: trusted artificial intelligence.

Денис Юрьевич ТУРДАКОВ – кандидат физико-математических наук, заведующий отделом ИСП РАН. Научные интересы: анализ социальных сетей, анализ текста, извлечение информации, обработка больших данных, методы доверенного искусственного интеллекта.

Denis Yurievich TURDAKOV – Cand. Sci. (Phys.-Math.), head of department ISP RAS. Research interests: social network analysis, text mining, information extraction, big data, trusted artificial intelligence.

DOI: 10.15514/ISPRAS-2025-37(6)-23

Применение контрастного обучения для семантической интерпретации русскоязычных таблиц

К.В. Тобола, ORCID: 0009-0006-1014-451X <kirilltobola@gmail.com>
Н.О. Дородных, ORCID: 0000-0001-7794-4462 <nikidorny@icc.ru>
Институт динамики систем и теории управления имени В.М. Матросова СО РАН,
Россия, 664033, г. Иркутск, ул. Лермонтова, д. 134.

Аннотация. Таблицы широко используются для представления и хранения данных, но, как правило, они не сопровождаются явной семантикой необходимой для машинной интерпретации своего содержания. Семантическая интерпретация таблиц является ключевой задачей для интеграции структурированных данных с графами знаний, однако существующие методы сталкиваются с проблемами при обработке русскоязычных таблиц из-за недостатка размеченных данных и языковой специфики. В данной работе предложен подход на основе контрастного обучения, направленный на устранение зависимости от ручной разметки и улучшение качества аннотирования столбцов редкими семантическими типами. Подход включает адаптацию алгоритма контрастного обучения для табличных данных с использованием аугментаций (удаление и перестановка ячеек), а также дистиллированной мультиязычной модели DistilBERT для эффективного обучения на неразмеченных данных корпуса RWT, содержащего 7.4 млн. столбцов. Обученные табличные представления интегрируются в конвейер аннотирования фреймворка RuTaBERT, что позволяет снизить вычислительные затраты. Эксперименты показали, что предложенный подход достигает микро-F1 97% и макро-F1 92%, превосходя некоторые базовые решения, что подтверждает его эффективность в условиях разреженности данных и языковых особенностей русского языка. Результаты демонстрируют, что контрастное обучение позволяет моделировать семантическое сходство между столбцами без явной разметки, что особенно важно для данных редких типов.

Ключевые слова: русскоязычные таблицы, табличные данные; семантическая интерпретация таблиц; семантическое аннотирование столбцов; графы знаний; самообучение; контрастное обучение; табличные представления.

Для цитирования: Тобола К.В., Дородных Н.О. Применение контрастного обучения для семантической интерпретации русскоязычных таблиц. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 107–122. DOI: 10.15514/ISPRAS–2025–37(6)–23.

Благодарности: Работа выполнена в рамках государственного задания Министерства науки и высшего образования Российской Федерации (тема № 1023110300006-9).

Using Contrastive Learning for Semantic Interpretation of Russian-Language Tables

K.V. Tobola, ORCID: 0009-0006-1014-451X <kirilltobola@gmail.com>
N.O. Dorodnykh, ORCID: 0000-0001-7794-4462 <nikidorny@icc.ru>

Matrosov Institute for System Dynamics and Control Theory of the Siberian Branch of Russian Academy of Sciences (ISDCT SB RAS),
134, Lermontov st., Irkutsk, 664033, Russia.

Abstract. Tables are widely used to represent and store data, but they are typically not accompanied by explicit semantics necessary for machine interpretation of their contents. Semantic table interpretation is critical for integrating structured data with knowledge graphs, but existing methods struggle with Russian-language tables due to limited labeled data and linguistic specificity. This paper proposes a contrastive learning-based approach to reduce dependency on manual labeling and improve column annotation quality for rare semantic types. The proposed approach adapts contrastive learning for tabular data using augmentations (removing/shuffling cells) and a distilled multilingual DistilBERT model trained on unlabeled RWT corpus (7.4M columns). The learned table representations are integrated into the RuTaBERT pipeline, which reduces computational costs. Experiments show micro-F1 0.974 and macro-F1 0.924, outperforming some baselines. This highlights the approach's efficiency in handling data sparsity and Russian language features. Results confirm that contrastive learning captures semantic column similarities without explicit supervision, crucial for rare data types.

Keywords: Russian-language tables, tabular data; semantic table interpretation; semantic column annotation; knowledge graphs; self-supervised learning; contrastive learning; table representations.

For citation: Tobola K.V., Dorodnykh N.O. Using contrastive learning for semantic interpretation of Russian-language tables. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 107-122 (in Russian). DOI: 10.15514/ISPRAS-2025-37(6)-23.

Acknowledgements. This work was supported by the state assignment of Ministry of Science and Higher Education of the Russian Federation (theme No. 1023110300006-9).

1. Введение

Табличные данные являются одним из ключевых форматов представления структурированной информации в различных областях: от научных исследований до бизнес-аналитики. Они используются в реляционных базах данных, электронных таблицах, веб-ресурсах и документах, что делает их обработку критически важной для автоматизации анализа данных. Однако таблицы, как правило, не сопровождаются явной семантикой необходимой для машинной интерпретации своего содержания. Поэтому семантическая интерпретация таблиц, особенно на языках отличных от английского, остается сложной задачей [1-2]. Основные вызовы связаны с необходимостью соотнесения отдельных элементов таблиц (столбцов, строк, ячеек) с понятиями из графов знаний, такими как DBpedia или Wikidata, а также с обработкой структурного и языкового разнообразия данных.

Русскоязычные таблицы представляют особую проблему из-за ограниченного количества специализированных инструментов и размеченных данных. Большинство современных методов, в частности, основанные на предварительно обученных языковых моделях по типу BERT [3-9], требуют огромных объемов размеченных данных, которые для русского языка часто недоступны или не сбалансированы. Кроме того, существующие решения, разработанные для английского языка, плохо адаптируются к другим языкам из-за различий в токенизации и контекстуальной семантике.

В данной работе предлагается новый подход для семантического аннотирования столбцов русскоязычных таблиц на основе контрастного обучения. Проверяется утверждение, что контрастное обучение на неразмеченных табличных данных улучшает способность модели различать семантические типы столбцов без использования вручную размеченного корпуса

табличных данных. Таким образом, подход позволяет эффективно использовать неразмеченные табличные данные для обучения устойчивых векторных представлений, снижая зависимость от ручной разметки. Наш вклад включает:

1) Адаптацию контрастного обучения для русскоязычных табличных данных с применением аугментаций. Обычно под аугментацией данных понимается техника искусственного увеличения размера обучающей выборки путем применения некоторых преобразований к исходным данным. Для табличных данных в этот процесс также включают удаление и перестановку ячеек.
2) Использование дистиллированной мультиязычной модели DistilBERT, что обеспечивает баланс между производительностью и вычислительными затратами.
3) Интеграцию предобученных табличных представлений в существующий конвейер аннотирования на базе фреймворка RuTaBERT [9], что демонстрирует гибкость подхода.
4) Эксперименты на крупномасштабном русскоязычном наборе данных RWT-RuTaBERT [10] показали, что предложенный подход превосходит некоторые базовые решения, что подтверждает его эффективность в условиях разреженности данных и языковой специфики.

Статья организована следующим образом: раздел 2 представляет современное состояние исследований в области семантической интерпретации таблиц. В разделе 3 описывается предложенный подход для семантического аннотирования столбцов русскоязычных таблиц, включая подготовку данных, архитектуру модели и алгоритм обучения. Раздел 4 содержит экспериментальные оценки тестирования производительности предлагаемого подхода. В заключении (раздел 5) дается обсуждение полученных результатов и планы будущей работы.

2. Современное состояние исследований

Под семантической интерпретацией (аннотированием) таблиц (*semantic table interpretation*) понимается процесс распознавания и связывания табличных данных с понятиями из некоторого целевого графа знаний, онтологии или другого внешнего словаря (например, DBpedia, Wikidata, Yago, Freebase, WordNet) [2, 11]. Одной из основных задач семантической интерпретации таблиц является аннотирование столбцов (*column type annotation*), при котором осуществляется сопоставление столбцов таблицы с семантическими типами (классами и свойствами) из целевого графа знаний.

За последние несколько лет существующие методы и модели использовали передовые достижения в области глубокого машинного обучения, формулируя задачу аннотирования столбцов как задачу классификации нескольких классов (*multi-class classification*). Так в работе [12] применяли нейронные сети и множество извлеченных групп признаков, таких как векторные представления слов и символов, а также глобальные статистики столбцов. В работе [13] добавлен анализ локального (внутри-табличного) контекста таблицы (соседних столбцов относительно целевого столбца), а в работе [14] добавился еще и межтабличный контекст для улучшения предсказаний. Однако особый интерес в данном контексте представляют работы, использующие предварительно обученные языковые модели на основе архитектуры трансформер (*Transformer*). Блоки трансформера используют механизм внимания, что позволяет модели генерировать полезные контекстуализированные векторные представления для структурных компонентов табличных данных, таких как ячейки, столбцы или строки. Также языковые модели, предварительно обученные на крупномасштабных текстовых корпусах, могут хранить семантику из обучающего текста в форме параметров модели, что делает процесс дообучения таких моделей на конкретных последующих задачах (*downstream tasks*) достаточно эффективным. Примерами таких работ являются модели TURL [3], TaPas [4], TaBERT [5], TABBIE [6], TUTA [7], Doduo [8].

Существующие решения в этой области имеют высокую производительность, которая достигается за счет большого количества размеченных обучающих данных. В частности, англоязычные наборы данных могут включать сотни тысяч размеченных столбцов (например, VizNet-Sato [13] ~ 100 000, WikiTables-TURL [3] ~ 600 000), а русскоязычный набор табличных данных RWT-RuTaBERT [10] насчитывает более 1.4 миллиона столбцов. Создание таких наборов является достаточно трудоемким процессом, требующим большого количества времени и ресурсов. Более того, для существующих наборов таблиц свойственна проблема разреженности данных, которая выражается в, довольно, несбалансированном распределении семантических типов (так называемое «*распределение с длинным хвостом*»). Так некоторым семантическим типам соответствуют сотни тысяч столбцов, а некоторым лишь несколько десятков. В результате модели сложно уловить достаточное количество сигналов для семантических типов, относящихся к меньшинству (редким типам, таким как «*атлет*», «*горный хребет*» или «*страховая компания*»), даже при контролируемых (*supervised*) настройках. Например, график распределения 30 наиболее встречающихся семантических типов для набора данных RWT-RuTaBERT, демонстрирующий эту проблему, представлен на рис. 1.

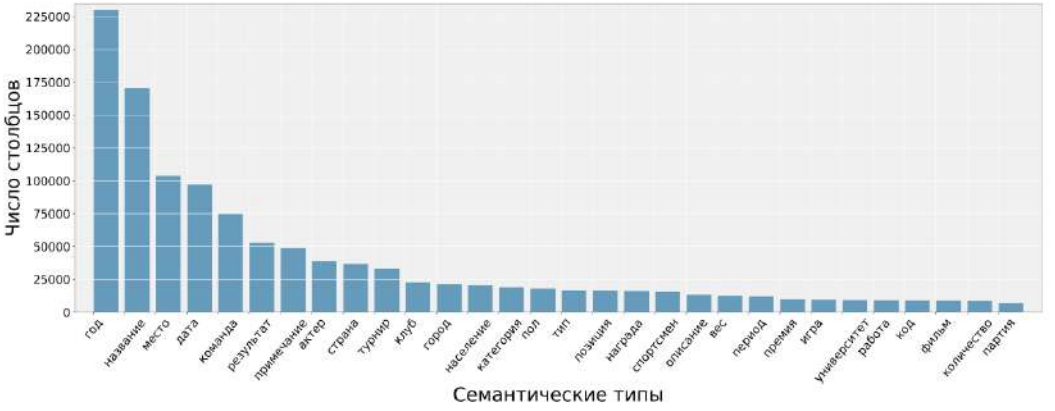


Рис. 1. Пример проблемы разреженности данных для набора RWT-RuTaBERT.
Fig. 1. Example of a data sparsity problem for the RWT-RuTaBERT dataset.

Существующие методы на основе предварительно обученных языковых моделей не являются универсальными. Наблюдается разрыв между эффективностью существующих решений на тестовых примерах и их применимостью на практике. Особенно это касается таблиц с различной языковой принадлежностью (представленных не на английском языке) и имеющих разную структурную компоновку.

Для повышения способности общего понимания таблиц и решения различных табличных задач появились работы, использующие большие языковые модели, которые часто имеют лучшую производительность по сравнению с предварительно обученными языковыми моделями, такими как BERT. Они также более устойчивы к невидимым примерам из-за специфичных эффектов, возникающих в результате размера модели, обученной на огромных объемах текста. Примерами таких работ являются модели Table-GPT [15] и TableLlama [16], а также подход [17]. Однако основным недостатком таких решений является то, что они требуют огромных вычислительных ресурсов, что затрудняет их практическое использование.

Для решения вышеуказанных проблем предлагается использовать методы самообучения (*self-supervised learning*), в частности, контрастного обучения (*contrastive learning*) для изучения табличных представлений, полученных на основе обширного корпуса неразмеченных табличных данных. Данные табличные представления могут быть

использованы как для определения родства (*relatedness*) между двумя таблицами (путем вычисления косинусного сходства между векторными представлениями), так и для тонкой настройки с дополнительными размеченными данными, содержащего небольшое количество таблиц, под конкретные последующие задачи (*downstream tasks*).

3. Предлагаемый подход

3.1 Постановка задачи

Таблица – это двухмерная структура данных, состоящая из строк и столбцов. В ячейках таблиц могут содержаться текстовые данные, числовые, дата и время и так далее. Можно выделить три вида таблиц с точки зрения структурированности информации:

1) *сильно структурированные* (таблицы реляционных баз данных);
2) *полуструктурированные* (электронные таблицы, составленные в специализированном программном обеспечении, например, MS Excel и так далее);
3) *неструктурированные* (изображения таблиц в документах формата PDF).

Те же таблицы можно классифицировать, вводя три основные группы в зависимости от ориентации:

1) *вертикальные* – таблицы, в которых данные расположены в виде вертикальных колонок (то есть идут "сверху вниз");
2) *горизонтальные* – таблицы, в которых данные расположены в виде горизонтальных линий (то есть идут "слева направо");
3) *матричные* – таблицы, в которых каждая запись индексируется ключом(ями) строки и ключом(ями) столбца.

В данной работе рассматриваются только вертикальные, сильно структурированные и полуструктурированные таблицы. Формальное описание входной таблицы можно представить как:

$$T = \{col_1, \dots, col_n\}, col_i = \{cell_1, \dots, cell_m\}, i \in \overline{1,n}, \quad (1)$$

где T – вертикальная таблица; col_i – i-столбец; $cell_j$ – j-ячейка i-столбца, при этом $j \in \overline{1,m}$.

Наша цель предсказать тип столбца, то есть классифицировать каждый столбец по его семантическому типу, например, «*Книга*», «*Писатель*», «*Жанр*» и «*Дата публикации*», вместо стандартных типов данных, таких как строка (*string*), число (*integer*) или дата (*datetime*). Предлагаемый подход подразумевает использование 170 различных семантических типов, которые были сформированы на основе отобранных классов и свойств (свойств-значений и объектных свойств) из графа знаний общего назначения DBpedia [18]. При этом брались только русские обозначения этих типов через метку языка (*label*), так как подход ориентирован на аннотирование русскоязычных таблиц. Формально данную задачу можно описать следующим образом:

$$P(col_i) \in KG_{st}, KG_{st} = \{st_1, \dots, st_{170}\}, \quad (2)$$

где $P(col_i)$ – предсказанный семантический тип для i-столбца; KG_{st} – множество всех семантических типов, мощность которого в данном случае равна 170.

Пример решения задачи аннотирования столбцов для входной таблицы представлен на рис. 2.

Основная идея подхода заключается в создании кодировщика устойчивых табличных представлений на основе контрастного обучения, которые затем можно использовать на последующих задачах (*downstream tasks*), в частности, для семантического аннотирования столбцов русскоязычных таблиц. Общая схема предлагаемого подхода представлена на рис. 3.

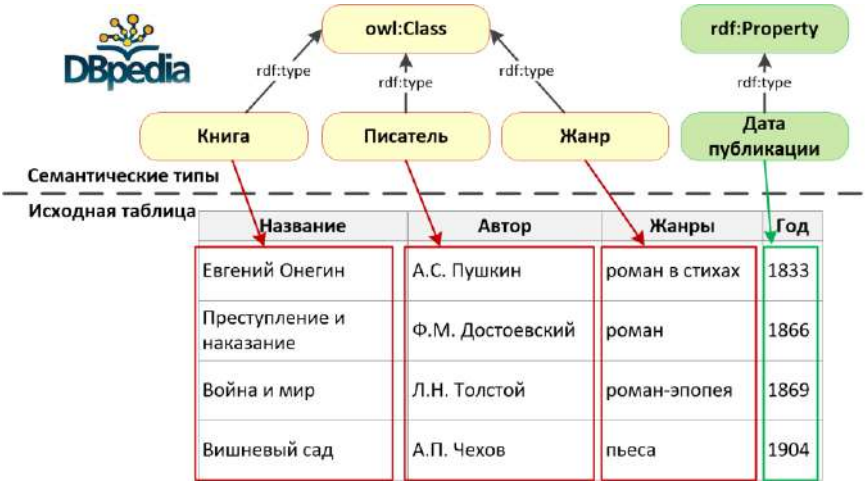


Рис. 2. Пример решения задачи аннотирования столбцов.
Fig. 2. Example of solving the column type annotation task.

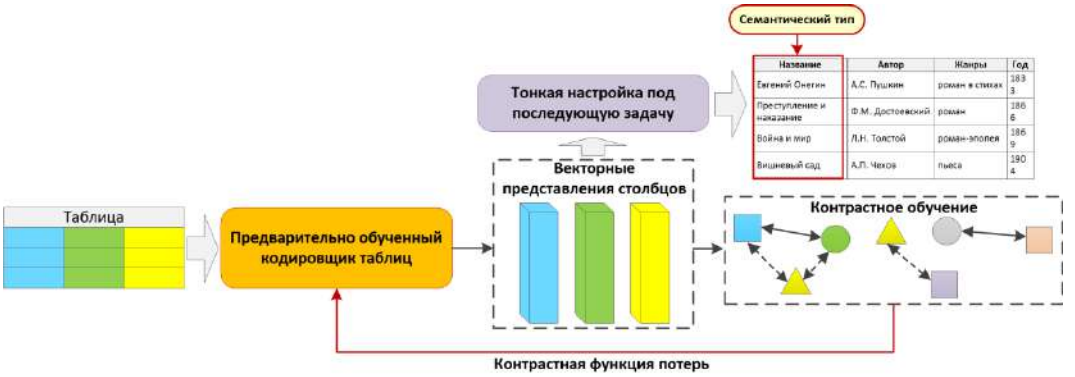


Рис. 3. Общая схема предлагаемого подхода.
Fig. 3. General outline of the proposed approach.

3.2 Описание набора данных

Предварительно обученный кодировщик таблиц (*Encoder*) обучается на огромном количестве табличных данных, которые не нуждается в ручной разметке. В качестве набора исходных таблиц используются крупномасштабный корпус RWT (Russian Web Tables) [19]. Данный набор представляет собой срез таблиц из русскоязычной Википедии за 13 сентября 2021 года. Основные статистики по набору RWT представлены в табл. 1.

На первом этапе предварительной обработки данных, из исходного корпуса RWT были отобраны вертикальные таблицы. При этом каждый столбец из такой таблицы представляется в качестве строки данных с помощью разделителя ячеек «<<». Пример хранения извлеченных столбцов приведен в табл. 2.

Далее была проведена очистка данных при помощи следующих операций:

- фильтрация пустых столбцов;
- удаление служебной информации парсера, оборачивающей текст при помощи регулярных выражений;
- удаление ссылок на статьи Википедии;

- удаление специальных символов (например, «@», «&», «?» и «!»);
- удаление пустых ячеек в столбце;
- удаление столбцов, имеющих длину меньше трех ячеек, так как данные столбцы после применения аугментаций удаления ячеек становятся не репрезентативными.

В результате всех операций очистки был получен набор неразмеченных русскоязычных табличных данных, состоящий из 4 656 668 столбцов.

Табл. 1. Статистика корпуса таблиц RWT.
Table 1. Statistics of the RWT table corpus.

Статистика	**Значение**
Число таблиц	1 266 731
Число столбцов	7 419 771
Число ячеек	99 638 194
Процент пустых столбцов	6%
Среднее количество ячеек в столбце	13.42

Табл. 2. Формат хранения извлеченных столбцов из корпуса RWT.
Table 2. Storage format for extracted columns from the RWT corpus.

id	**table_id**	**column_id**	**column_header**	**column_data**
0	7545708	0	Название	Сан-Хуан (исп. San Juan) << Вальe-Нуэво...
1	5433710	3	Зрители	100 << 200 << 50 << 300 << 500
…	…	…	…	…

В проведенном эксперименте предварительная обработка таблиц осуществлялась в автоматизированном режиме с использованием специального средства [9].

3.3 Алгоритм обучения

Контрастное обучение (*contrastive learning*) – это одна из техник самообучения, предназначенная для получения информативных векторных представлений. Она заключается в максимизации некоторой метрики согласованности, в нашем случае косинусного сходства, между положительными парами (экземплярами данных) и минимизации данной метрики между отрицательными парами. Контрастное обучение позволяет эффективно обучаться на неразмеченном корпусе данных.

В данной работе адаптируется концепция контрастного обучения, предложенная в работе [20], для табличных данных. Алгоритм контрастного обучения для табличных данных представлен на рис. 4.

Основная идея заключается в построении во время обучения для каждого столбца в пакете данных двух аугментаций. Для полученных аугментаций формируются векторные представления при помощи модели кодировщика. Векторные представления для аугментаций, полученные на одном столбце, считаются положительной парой, наша задача максимизировать метрику косинусного сходства для этой пары. В свою очередь векторные представления аугментаций, построенных для различных столбцов, считаются отрицательными парами. Для этих пар решается задача минимизации метрики косинусного сходства.

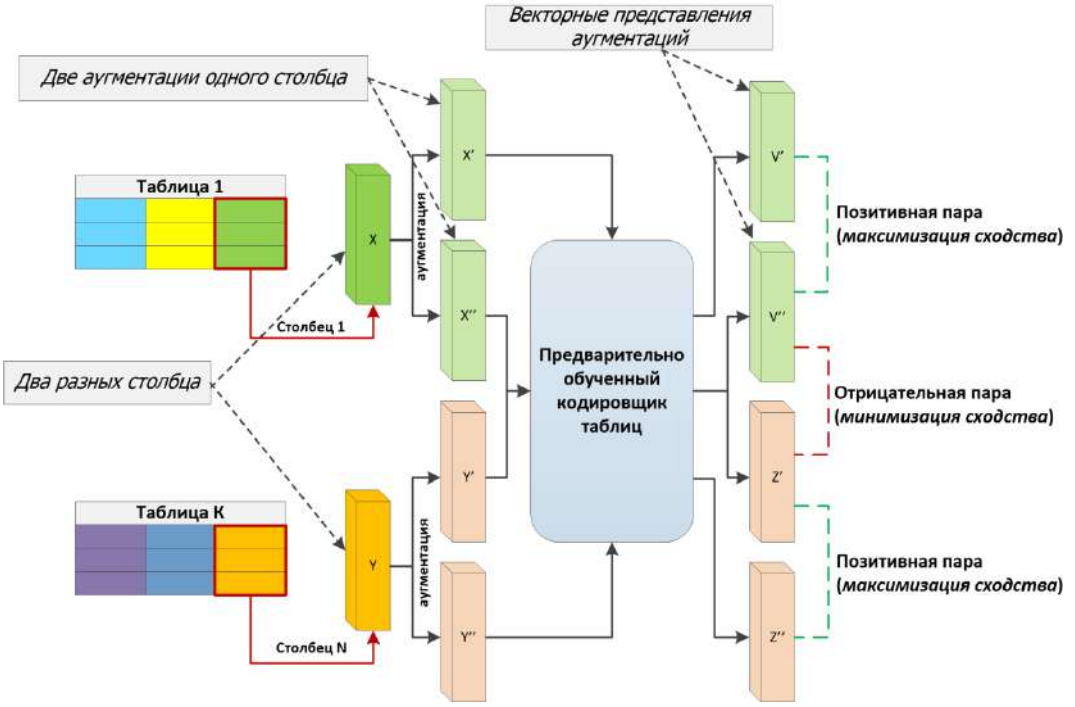


Рис. 4. Алгоритм контрастного обучения для табличных данных.
Fig. 4. Contrastive learning algorithm for tabular data.

3.3.1 Аугментация данных

Техника аугментации данных широко используется в условиях ограниченного количества размеченных данных или их отсутствии для увеличения обобщающей способности модели. В контрастном обучении аугментации играют определяющую роль в формировании семантически согласованных положительных пар.

Как правило, для табличных данных выделяют следующие аугментации:

- удаление случайной ячейки;
- удаление/перестановка/замена токенов в ячейке;
- выборка строк (например, в размере 50%);
- перестановка ячеек в строке таблицы;
- удаление столбцов;
- перестановка столбцов в таблице;

На данный момент нет исследований по определению аугментаций, работающих наилучшим образом для формирования семантически согласованных пар в контексте обработки табличных данных. Поэтому в данной работе, были выбраны две аугментации, которые по нашему предположению, являются наиболее перспективными, а именно: удаление случайных ячеек и перестановка ячеек в столбце. При удалении случайных ячеек, удаляются 10% от всех ячеек в столбце.

3.3.2 Контрастная функция потерь

В задачах обучения представления активно используют контрастные функции потерь (*contrastive loss*), так как с их помощью модель способна лучше различать внутренние

структуры данных, и как следствие лучше извлекать полезные представления. Контрастная функция потерь направлена на максимизацию согласования между положительными парами и минимизацию согласования между отрицательными парами в векторном пространстве.

Существует несколько вариаций для контрастных функций потерь. В данной работе выбрана функция NT-Xent loss (Normalized Temperature-Scaled Cross-Entropy Loss), используемая в работе [20]. Данная функция определяется следующим образом:

$$L_{NT-Xent} = \frac{1}{2N} \times \sum_{k=1}^{N} [l(2k-1,2k) + l(2k,2k-1)],$$

$$l(i,j) = -\log \frac{exp\left(\frac{s(i,j)}{\tau}\right)}{\sum_{k=1}^{2N} 1_{[k \neq i]} exp\left(\frac{s(i,k)}{\tau}\right)},$$

$$s(i,j) = \frac{z_i \times z_j}{\|z_i\| \times \|z_j\|}, \quad (4)$$

где $1_{[k \neq i]} \in \{0,1\}$ принимает значение 1, если $k \neq i$, иначе 0; τ – параметр температуры; $s(i,j)$ – косинусное сходство; $z_i = [z_1, \dots, z_{256}], z \in \mathbb{R}$ – векторное представление (выход модели).

3.4 Архитектура модели

В настоящий момент модели архитектуры трансформер являются ключевыми в решении задач обработки естественного языка. Данные модели являются универсальными инструментами для решения задач обработки текстов за счет их возможности учитывать контекстные зависимости между словами в последовательностях, а также возможности обучаться на неразмеченных или частично размеченных данных. Причем они делают это достаточно эффективно, за счет высокого параллелизма, что делает их предпочтительными для обучения на больших объемах данных.

Согласно работе [20] двумя ключевыми гиперпараметрами контрастного обучения являются размер пакета данных и количество эпох. Чем больше размер пакета данных и количество эпох, тем более репрезентативными получаются векторные представления обучаемой модели, тем лучше она показывает себя на последующих конкретных задачах при тонкой настройке.

Исходя из этого, в качестве базовой модели кодировщика была взята дистиллированная мультиязычная модель BERT [21]. Данная модель обучена на корпусе статьей из Википедии, написанных на 104 различных языках. В отличии от базовой версии [22] данная модель состоит всего из 6 слоев, что в два раза меньше, чем у базовой версии, и 12 модулей механизма внимания. Данная модель насчитывает 134 миллиона параметров (у базовой версии 177 миллионов параметров).

Дистилляция моделей машинного обучения – это техника, при которой осуществляется перенос знаний от более сложной модели, также называемой учителем, к более компактной, называемой учеником. При этом сохраняется качество предсказаний модели.

Данная техника в сочетании с уменьшением максимальной длины последовательности токенизатора, инструмента для разбиения текстового документа на более мелкие единицы (токены), до 256 токенов, позволила обучить модель с размером пакета данных, равным 800, это в 25 раз больше, чем у аналогичного передового англоязычного решения [23].

В адаптируемой работе [20] было проведено исследование об использовании проекции выходного слоя кодировщика в некоторое латентное векторное пространство, в котором рассчитывается контрастная функция потерь. Результаты показывают, что применение нелинейной проекции во время обучения положительно влияет на качество представлений. Таким образом, в данной работе используется двухслойный перцептрон (Multilayer

perceptron, MLP), идущий после выходного слоя кодировщика, для проекции в латентное векторное пространство, размерности 128, в котором рассчитывается контрастная функция потерь по указанным выше формулам.

4. Экспериментальные оценки и обсуждение

Все эксперименты проводились на кластере «Академик В.М. Матросов» на базе Института Динамики систем и Теории управления СО РАН (ИДСТУ СО РАН). В конфигурацию кластера входят два 16-ти ядерных процессора Intel Xeon Gold 6326 «Ice Lake» 2.9 GHz, четыре графических процессора NVIDIA A100 80 GB PCIe и 2 TB оперативной памяти DDR4-3200.

4.1 Настройки контрастного обучения

Подход реализован на языке Python с использованием библиотек PyTorch и Transformers. В качестве оптимизатора градиентного спуска был выбран классический метод AdamW (lr = 5–e5, eps = 1e-6). Для ускорения сходимости модели была применена техника косинусного отжига (*cosine annealing*), предназначенная для динамического уменьшения скорости обучения (*learning rate*). Параметр температуры, являющийся гиперпараметром контрастной функции потерь, равен значению 0.1, так как данное значение является оптимальным, согласно работе [20].

В таких настройках модель предварительно обученного кодировщика была обучена на протяжении 100 эпох, на четырех графических ускорителях NVIDIA A100 с использованием технологии Distributed-Data-Parallel фреймворка PyTorch. Обучение продлилось на протяжении 9 дней 9 часов 53 минут. Количество потребляемой памяти графического процессора составило 290 гигабайт.

4.2 Настройки модели для семантического аннотирования столбцов

В настоящей работе в качестве последующей конкретной задачи выступает семантическая интерпретация (аннотирование) столбцов таблиц. Для решения этой задачи, ранее в работе [9] был предложен фреймворк RuTaBERT, основанный на тонкой настройке предварительно обученной мультиязычной языковой модели BERT с использованием специально подготовленного набора таблиц RWT-RuTaBERT [10]. Данный набор содержит примерно 1.56 миллиона размеченных столбцов. Основная идея заключается в том, чтобы использовать уже готовый конвейер (*pipeline*) этого фреймворка с заменой стандартной модели BERT на специализированный предварительно обученный табличный кодировщик. При этом в качестве набора данных для обучения также выступает размеченный ранее набор RWT-RuTaBERT со всеми стандартными настройками. Размер валидационной выборки составил 5% от общего числа тренировочного подмножества. В качестве разложения значений столбцов в последовательности токенов используется техника сериализации соседних столбцов (*neighboring column serialization*), также предложенная в работе [9].

Согласно работе [20], слой проекции обучен быть инвариантным к преобразованию данных, из-за чего он может потерять информацию, которая может быть полезна для последующих задач (*downstream tasks*). Поэтому для дальнейшего дообучения табличного кодировщика использовался выход с первого линейного слоя проекции с применением функции активации LeakyReLU. При этом применялись стандартные настройки обучения, заданные во фреймворке RuTaBERT. Таким образом, модель была дообучена на протяжении 30 эпох с размером пакета данных равным 32 на наборе данных RWT-RuTaBERT. Для этого использовались 2 графических ускорителя NVIDIA A100, обучение продлилось на протяжении 2 дней 20 часов 15 минут. Потребление памяти графического процессора составило 9.9 гигабайт. Дополнительно была обучена модель с размером пакета данных равным 256 с сохранением всех остальных гиперпараметров. При таких настройках обучение

заняло 4 дня 3 часа 1 минуту. Количество потребляемой памяти графического процессора составило 52 гигабайта.

4.3 Метрики оценки

В качестве основных метрик оценки производительности предлагаемого подхода выступает усредненная F1-мера, поскольку решается задача классификации нескольких классов (*multi-class classification*). В частности, используется микро F1-мера (*micro F1*), макро F1-мера (*macro F1*) и взвешенная F1-мера (*weighted F1*) так как набор данных RWT-RuTaBERT не сбалансирован.

Микро F1-мера рассчитывается по всей матрице ошибок и определяется следующим образом:

$$MicroPrecision = \frac{TP_1+TP_2+\cdots+TP_n}{TP_1+\cdots+TP_n+FP_1+\cdots+FP_n}, \quad MicroRecall = \frac{TP_1+TP_2+\cdots+TP_n}{TP_1+\cdots+TP_n+FN_1+\cdots+FN_n},$$

$$MicroF1 = \frac{2\times MicroPrecision\times MicroRecall}{MicroPrecision+MicroRecall}, \tag{5}$$

Макро F1-мера – это средняя оценка F1-меры для каждого семантического типа (класса). При этом все классы являются эквивалентными, то есть не учитывается дисбаланс классов. Макро F1-мера рассчитывается по формуле:

$$MacroF1 = \frac{\sum_{i=1}^{N} F1_i}{N}, \tag{6}$$

где N – число семантических типов (классов); $F1_i$ – F1-мера для i-семантического типа.

Взвешенная F1-мера рассчитывается для каждого класса и затем суммируется как средневзвешенное значение с учетом количества записей для каждого класса. Данная метрика в отличие от микро F1-меры учитывает дисбаланс классов. Взвешенная F1-мера рассчитывается по формуле:

$$WeightedF1 = \sum_{i=1}^{C}[w_i \times F1_i], w_i = \frac{n_i}{N}, \tag{7}$$

где C – число семантических типов (классов); n_i – количество экземпляров в i-классе; N – общее количество экземпляров; $F1_i$ – F1-мера для i-семантического типа.

4.4 Результаты и обсуждение

Результаты экспериментальной оценки представлены в табл. 3. При этом осуществлено сравнение производительности предлагаемого подхода с некоторыми аналогами, которые выбраны в качестве базовых решений (*baseline*).

Во-первых, выбрана предварительно обученная языковая модель RuBERT [24], которая специализируется на обработке русского языка. При этом применена одна из техник трансферного обучения (*transfer learning*), при которой в процессе обучения веса слоев кодировщика остаются неизменными. Таким образом, во время дообучения RuBERT на наборе данных RWT-RuTaBERT изменялись только параметры слоя, предназначенного для классификации данных.

Во-вторых, выбран современный фреймворк Doduo [8], который является лидером решения задачи семантического аннотирования столбцов и связей между ними. Таким образом, в данном случае также применялась техника трансферного обучения с заморозкой слоев кодировщика и дообучением только последнего линейного слоя классификатора (*Doduo*). Помимо этого, было рассмотрено альтернативное базовое решение (*Translated Doduo*), основанное на переводе русскоязычного набора табличных данных на английский язык и его применения для англоязычной модели Doduo. Таким образом, семантические типы RWT-RuTaBERT были отображены в английские типы набора данных VizNet-Sato (например,

«*год*»: «*year*», «*место*»: «*location*», «*альбом*»: «*album*» и так далее). При отображении семантических типов, 16 типов из VizNet-Sato были исключены, в связи с невозможностью их отображения («*affiliate*», «*birthPlace*», «*brand*», «*collection*», «*command*», «*currency*», «*family*», «*fileSize*», «*gender*», «*grades*», «*isbn*», «*jockey*», «*plays*», «*sales*», «*species*», «*teamName*».). В свою очередь из набора RWT-RuTaBERT 5 семантических типов не получилось отобразить ни в один тип VizNet-Sato («*площадь*», «*звук*», «*отношение*», «*ввп*», «*лес*»). Сами столбцы из тестового набора RWT-RuTaBERT были переведены на английский язык с помощью машинного перевода. Кроме того, была осуществлена полноценная тонкая настройка мультиязычной модели BERT согласно подходу Doduo на наборе данных RWT-RuTaBERT (*Fine-tuned Doduo*).

В-третьих, взят оригинальный подход RuTaBERT из работы [9].

Табл. 3. Результаты экспериментальной оценки на наборе данных RWT-RuTaBERT.
Table 3. The results of the experimental evaluation on the RWT-RuTaBERT dataset.

Подход	**micro F1**	**macro F1**	**weighted F1**
Doduo	0.140	0.043	—
Translated Doduo	0.364	0.127	—
RuBERT	0.612	0.417	0.592
Fine-tuned Doduo	0.962	0.891	0.962
RuTaBERT	0.964	0.904	0.963
Предлагаемый подход (bs32)	0.969	0.910	0.969
Предлагаемый подход (bs256)	**0.974**	**0.924**	**0.974**

Полученная оценка показала, что предлагаемый подход в обеих конфигурациях обучения (размер пакета данных 32 и 256) превзошел все базовые решения. В частности, эксперимент продемонстрировал, что модель RuBERT хоть и ориентирована на обработку русского языка, но не направлена напрямую на решение табличных задач, которые оказались сложны для этой модели. Таким образом, существующие русскоязычные модели не могут быть эффективно применены к решению задачи семантического аннотирования столбцов.

Модель Doduo обученная с применением техники трансферного обучения показала достаточно низкие результаты оценки. Это связано с тем, что модель обучалась на табличных данных, представленных только на английском языке. В частности, в токенизаторе этой модели практически отсутствуют русскоязычные токены. Вследствие чего можно прийти к выводу, что невозможно взять модель, обученную на английском языке, и использовать ее на некотором другом языке, в данном случае русском. Для этого необходимо менять базовый кодировщик, способный различать этот самый язык.

При этом тонко настроенный мультиязычный кодировщик фреймворка Doduo и подход RuTaBERT показали почти сопоставимые результаты по метрикам оценки. Однако можно заметить, что использование предварительно обученного кодировщика таблиц на основе контрастного обучения положительно влияет на результат. С использованием меньшей модели при тех же настройках получилось достичь точно таких же результатов, как у классической модели RuTaBERT и тонко настроенной Doduo. При этом модель потребляет примерно в 3 раза меньше видеопамяти в процессе обучения, менее 10 гигабайт (при одинаковом для всех трех моделей размере пакета данных равным 32), что позволяет запускать обучение на стационарном домашнем компьютере. Кроме того, при использовании

большего размера пакета данных (256, в отличие от 32), получилось достичь выигрыш в 1.5% относительно классической модели RuTaBERT и почти 3% по сравнению с тонко настроенной Doduo. Полученные экспериментальные результаты указывают на потенциал нашего подхода для семантической аннотации русскоязычных таблиц.

Для дальнейшей оценки эффективности предлагаемого подхода, был проведен статистический анализ по трем аспектам:

1) Группировка по типам данных: Все столбцы из собранных таблиц были классифицированы на 5 основных групп: «*Дата*», «*Число*», «*Ссылка*», «*Короткий текст*» и «*Длинный текст*». Столбцы с типом «*Дата*» включают даты, годы или время в различных форматах. Столбцы типа «*Число*» содержат только числа, например, результаты измерений длины, веса или возраста. Столбцы с типом «*Ссылка*» включают различные виды ссылок, включая URL-адреса. Текстовые столбцы были разделены на «*Короткий текст*» (значение ячейки содержит менее четырех лексем) и «*Длинный текст*» (значение ячейки содержит четыре и более лексем). Также был выделен отдельный тип данных «*Персона*», который учитывает распространенность таких семантических типов как «*работодатель*», «*сценарист*», «*спортсмен*», «*футболист*» и так далее. В табл. 4 представлены результаты микро F1-меры для каждой группы типов данных.

Табл. 4. Результаты экспериментальной оценки по микро F1 для выделенных типов данных.
Table 4. The results of the experimental evaluation of micro F1 for selected data types.

Тип данных	RuTaBERT	Предлагаемый подход
Дата	0.941	**0.948**
Длинный текст	**0.885**	0.858
Число	0.749	**0.760**
Персона	0.692	**0.716**
Короткий текст	0.926	**0.932**
Ссылка	**0.699**	0.611

2) Сходимость модели: Для оценки сходимости модели были проведены эксперименты для отдельных контрольных точек (*checkpoints*) обученной модели (*bs32*) в сравнение с моделью RuTaBERT. При этом брались две контрольных точки моделей, обученных в течение 10 и 30 эпох. Результаты представлены в табл. 5. В скобках представлено значение прироста оценки относительно первой контрольной точки.

Можно заметить, что модель, обученная по предлагаемому подходу, сходится быстрее, чем модель RuTaBERT, при этом имеет на 1-3% выше производительность на тестовой выборке по метрике F1. Данная особенность позволяет использовать меньшее количество эпох на этапе обучения, получая при этом сравнимую или даже превосходящую производительность в сравнении с моделью RuTaBERT.

3) Редкие семантические типы: Оценка производительности также проводилась для 15 наименее встречаемых семантических типов. Для сравнения были использованы контрольные точки обученной модели (bs32) и RuTaBERT, достигшие наивысшего результата на обучающем наборе данных по метрике макро F1. Результаты представлены в табл. 6. Результаты показывают, что благодаря полученным устойчивым табличным представлениям, предлагаемый подход значительно превосходит существующее современное русскоязычное решение, в частности, RuTaBERT по производительности, в особенности для редко встречающихся семантических типов.

Табл. 5. Результаты экспериментальной оценки сходимости моделей.
Table 5. The results of an experimental assessment of the convergence of the models.

Модель	micro F1	macro F1	weighted F1
RuTaBERT (10 эпох)	0.952	0.856	0.952
Предлагаемый подход (10 эпох)	0.966	0.888	0.966
RuTaBERT (30 эпох)	0.964 (+0.012)	0.904 (+0.048)	0.963 (+0.011)
Предлагаемый подход (30 эпох)	0.969 (+0.003)	0.910 (+0.022)	0.969 (+0.003)

Табл. 6. Результаты экспериментальной оценки по микро F1 для 15 редко встречающихся семантических типов.
Table 6. The results of an experimental evaluation of micro F1 for 15 rarely occurring semantic types.

Семантический тип	Кол-во вхождений (в тестовой выборке)	RuTaBERT	Предлагаемый подход
камера	102 (4)	0.250	0.75
работодатель	101 (10)	0.899	1.000
устройство	101 (8)	0.625	0.875
животное	93 (7)	0.857	1.000
журнал	93 (9)	0.440	0.440
континент	92 (8)	0.625	0.750
роман	89 (11)	0.818	0.909
закон	89 (9)	1.000	1.000
борец	88 (5)	0.400	0.600
колледж	87 (5)	0.000	0.200
музей	86 (4)	0.500	0.750
фирма	85 (6)	0.333	0.333
префектура	83 (10)	0.600	0.699
дорога	83 (6)	0.500	0.666
цитата	76 (7)	0.857	1.000

5. Заключение

В работе предложен подход семантического аннотирования столбцов русскоязычных таблиц, основанный на контрастном обучении. Подход реализован в форме инструментального средства. Исходный код [25] и обученная модель [26] опубликованы в открытом доступе. Экспериментальные результаты демонстрируют, что подход устраняет зависимость от

больших объемов размеченных данных за счет самообучения на неразмеченных таблицах. При этом он превосходит существующие базовые решения (Doduo и RuTaBERT) по метрикам оценки, особенно для редких семантических типов. Подход также обеспечивает вычислительную эффективность за счет использования дистиллированной модели, что снижает требования к памяти на 60% по сравнению с аналогами.

Полученные результаты экспериментальной оценки показали эффективность предлагаемого решения. В будущем, в рамках исследовательского проекта с Институтом системного программирования имени В.П. Иванникова Российской академии наук (ИСП РАН), планируется интегрировать эти результаты в виде специального обработчика таблиц, входящего в состав платформы Talisman [27]. Также планируется расширение подхода на таблицы с горизонтальной и матричной компоновкой. Дополнительно будет исследоваться вопрос использования новых аугментаций для улучшения устойчивости табличных представлений.

В целом предлагаемый подход открывает возможности для создания универсальных систем семантической интерпретации таблиц, что актуально для задач интеграции различной структурированной и слабоструктурированной информации, и бизнес-аналитики.

Список литературы

[1]. Badaro G., Saeed M., Papotti P. Transformers for Tabular Data Representation: A Survey of Models and Applications. Transactions of the Association for Computational Linguistics, vol. 11, 2023, pp. 227-249. DOI: 10.1162/tacl_a_00544.

[2]. Liu J., Chabot Y., Troncy R., Huynh V.-P., Labbe T., Monnin P. From tabular data to knowledge graphs: A survey of semantic table interpretation tasks and methods. Journal of Web Semantics, vol. 76, 2023, 100761. DOI: 10.1016/j.websem.2022.100761.

[3]. Deng X., Sun H., Lees A., Wu Y., Yu C. TURL: Table Understanding through Representation Learning. Proc. the VLDB Endowment, vol. 14, no. 3, 2020, pp. 307-319. DOI: 10.14778/3430915.3430921.

[4]. Herzig J., Nowak P. K., Muller T., Piccinno F., Eisenschlos J. M. TaPas: Weakly Supervised Table Parsing via Pre-training. Proc. 58th Annual Meeting of the Association for Computational Linguistics, Online, 2020, pp. 4320-4333. DOI: 10.18653/v1/2020.acl-main.398.

[5]. Yin P., Neubig G., Yih W. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. Proc. the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 8413-8426. DOI: 10.18653/v1/2020.acl-main.745.

[6]. Iida H., Thai D., Manjunatha V., Iyyer M. TABBIE: Pretrained Representations of Tabular Data. Proc. the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021, pp. 3446-3456. DOI: 10.18653/v1/2021.naacl-main.270.

[7]. Wang Z., Dong H., Jia R., Li J., Fu Z., Han S., Zhang D. TUTA: Tree-based Transformers for Generally Structured Table Pre-training. Proc. the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD'21), 2021, pp. 1780-1790. DOI: 10.1145/3447548.3467434.

[8]. Suhara Y., Li J., Li Y. Annotating Columns with Pre-trained Language Models. Proc. the 2022 International Conference on Management of Data (SIGMOD'22), 2022, pp. 1493-1503. DOI: 10.1145/3514221.3517906.

[9]. Tobola K. V., Dorodnykh N. O. Semantic Annotation of Russian-Language Tables Based on a Pre-Trained Language Model. Proc. the 2024 Ivannikov Memorial Workshop (IVMEM), Velikiy Novgorod, Russian Federation, 2024, pp. 62-68. DOI: 10.1109/IVMEM63006.2024.10659709.

[10]. RWT-RuTaBERT, Available at: https://huggingface.co/datasets/sti-team/rwt-rutabert, accessed 06.05.2025.

[11]. Ji S., Pan S., Cambria E., Marttinen P., Yu P.S. A Survey on Knowledge Graphs: Representation, Acquisition and Applications. IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 2, 2021, pp. 494-514. DOI: 10.1109/TNNLS.2021.3070843.

[12]. Hulsebos M., Hu K., Bakker M., Zgraggen E., Satyanarayan A., Kraska T., Demiralp Ç., Hidalgo C. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. Proc. the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'19), 2019, pp. 1500-1508. DOI: 10.1145/3292500.3330993.

[13]. Zhang D., Suhara Y., Li J., Hulsebos M., Demiralp C., Tan W.-C. Sato: Contextual semantic type detection in tables. Proc. the VLDB Endowment, vol. 13, no. 11, 2020, pp. 1835-1848. DOI: 10.14778/3407790.3407793.
[14]. Wang D., Shiralkar P., Lockard C., Huang B., Dong X. L., Jiang M. TCN: Table Convolutional Network for Web Table Interpretation. Proc. the Web Conference (WWW'21), Ljubljana, Slovenia, 2021, pp. 4020-4032. DOI: 10.1145/3442381.3450090.
[15]. Li P. , He Y., Yashar D., Cui W., Ge S., Zhang H., Fainman D. R., Zhang D., Chaudhuri S. Table-GPT: Table Fine-tuned GPT for Diverse Table Tasks. Proceedings of the ACM on Management of Data, vol. 2, no. 3, 2024, pp. 1-28. DOI: 10.1145/3654979.
[16]. Zhang T., Yue X., Li Y., Sun H. TableLlama: Towards Open Large Generalist Models for Tables. Proc. the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Mexico City, Mexico, 2024, pp. 6024-6044. DOI: 10.18653/v1/2024.naacl-long.335.
[17]. Korini K., Bizer C. Column Property Annotation Using Large Language Models. Proc. the Semantic Web: ESWC 2024 Satellite Events, Hersonissos, Crete, Greece, 2024, pp. 61-70. DOI: 10.1007/978-3-031-78952-6_6.
[18]. DBpedia, Available at: https://www.dbpedia.org/, accessed 06.05.2025.
[19]. Ru-Wiki-Tables-dataset, Available at: https://gitlab.com/unidata-labs/ru-wiki-tables-dataset, accessed 06.05.2025.
[20]. Chen T., Kornblith S., Norouzi M., Hinton G. A simple framework for contrastive learning of visual representations. Proc. the 37th International Conference on Machine Learning (ICML'20), Online, 2020, pp. 1597-1607. DOI: 10.5555/3524938.3525087.
[21]. Model Card for DistilBERT base multilingual (cased), Available at: https://huggingface.co/distilbert/distilbert-base-multilingual-cased, accessed 06.05.2025.
[22]. BERT multilingual base model (cased), Available at: https://huggingface.co/google-bert/bert-base-multilingual-cased, accessed 06.05.2025.
[23]. Miao Z., Wang J. Watchog: A Light-weight Contrastive Learning based Framework for Column Annotation. Proceedings of the ACM on Management of Data, vol. 1, no. 4, 2023, pp. 1-24. DOI: 10.1145/3626766.
[24]. Kuratov Y., Arkhipov M. Adaptation of Deep Bidirectional Multilingual Transformers for Russian Language, 2019, arXiv:1905.07213.
[25]. CoLeM framework, Available at: https://github.com/YRL-AIDA/CoLeM, accessed 06.05.2025.
[26]. CoLeM base cased, Available at: https://huggingface.co/sti-team/colem-base-cased, accessed 06.05.2025.
[27]. Talisman, Available at: http://talisman.ispras.ru, accessed 06.05.2025.

Информация об авторах / Information about authors

Кирилл Владимирович ТОБОЛА – аспирант Института динамики систем и теории управления им. В.М. Матросова Сибирского отделения РАН (ИДСТУ СО РАН) с 2024 года. Сфера научных интересов: табличные представления, большие языковые модели для реляционных данных, извлечение данных из табличных источников.

Kirill Vladimirovich TOBOLA is a postgraduate student at the Matrosov Institute of System Dynamics and Control Theory named SB RAS (ISDCT SB RAS) since 2024. Research interests: table embedding, large language models for relational data, and data extraction from tabular sources.

Никита Олегович ДОРОДНЫХ – кандидат технических наук, старший научный сотрудник Института динамики систем и теории управления им. В.М. Матросова Сибирского отделения РАН (ИДСТУ СО РАН) с 2021 года. Сфера научных интересов: автоматизация создания интеллектуальных систем и баз знаний, получение знаний на основе преобразования концептуальных моделей и электронных таблиц.

Nikita Olegovych DORODNYKH is PhD, senior associate researcher at the Matrosov Institute of System Dynamics and Control Theory named SB RAS (ISDCT SB RAS) since 2021. Research interests: computer-aided development of intelligent systems and knowledge bases, knowledge acquisition based on the transformation of conceptual models and tables.

DOI: 10.15514/ISPRAS-2025-37(6)-24

Сравнение классических и нейросетевых алгоритмов выделения ключевых точек на изображениях пересеченной местности для применения в SLAM алгоритмах

П.А. Ухов, ORCID: 0000-0002-3728-2262 <ukhov79@gmail.com>
М.Б. Булакина, ORCID: 0009-0008-8354-0294 <maria.b.bulakina@gmail.com>
С.С. Крылов, ORCID: 0000-0003-3267-6411 <compgra@yandex.ru>

Московский авиационный институт (национальный исследовательский университет), Россия, 125993, г. Москва, Волоколамское шоссе, д. 4.

Аннотация. В работе предложена метрика для оценки качества работы алгоритмов выделения ключевых точек на изображении в условиях пересеченной местности при отсутствии однозначно определяемых ориентиров и углов. Проведено сравнение различных алгоритмов выделения ключевых точек для последующей реализации в составе SLAM алгоритма на борту беспилотного летательного аппарата. Получены значения предложенной метрики для популярных алгоритмов выделения ключевых точек и другие параметры на основе запуска решения в контролируемом окружении. Показаны преимущества алгоритмов на основе моделей машинного обучения.

Ключевые слова: алгоритмы выделения ключевых точек на изображении; алгоритмы SLAM; алгоритм SIFT; алгоритм SURF; алгоритм ORB; алгоритм SuperPoint; алгоритм R2D2; метод LightGlue; машинное обучение.

Для цитирования: Ухов П.А., Булакина М.Б., Крылов С.С. Сравнение классических и нейросетевых алгоритмов выделения ключевых точек на изображениях пересеченной местности для применения в SLAM алгоритмах. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 123–130. DOI: 10.15514/ISPRAS–2025–37(6)–24.

Comparison of Classical and Machine Learning Algorithms for Feature Point Extraction in Rugged Terrain Images for Application in SLAM Algorithms

P.A. Ukhov, ORCID: 0000-0002-3728-2262 <ukhov79@gmail.com>
M.B. Bulakina, ORCID: 0009-0008-8354-0294 <maria.b.bulakina@gmail.com>
S.S. Krylov, ORCID: 0000-0003-3267-6411 <compgra@yandex.ru>

Moscow Aviation Institute (National Research University)
4, Volokolamskoe highway, Moscow, 125993, Russia.

Abstract. The paper proposes a metric for evaluating the performance of feature point extraction algorithms in rough terrain conditions with no clearly defined landmarks or corners. Various feature point detection algorithms are compared for subsequent integration into a SLAM algorithm on board an unmanned aerial vehicle (UAV). The proposed metric, along with other algorithm parameters, is evaluated through experiments conducted in a controlled environment. The advantages of algorithms based on machine learning models are demonstrated.

Keywords: feature point detection algorithms; SLAM; SIFT; LightGlue; SURF; ORB; SuperPoint; R2D2; machine learning.

For citation: Ukhov P.A., Bulakina M.B., Krylov S.S. Comparison of classical and machine learning algorithms for feature point extraction in rugged terrain images for application in SLAM algorithms. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 123-130 (in Russian). DOI: 10.15514/ISPRAS-2025-37(6)-24.

1. Введение

В последнее время с учетом недоступности глобальных навигационных систем в ряде регионов Российской Федерации и увеличившимися рисками для беспилотных летательных аппаратов (далее БПЛА) в связи с нарушением навигационных полей все большее значение приобретают задачи автономной навигации. Для БПЛА небольшого размера на текущем уровне развития технологий инерциальных навигационных систем невозможно обеспечение требуемой точности для выполнения миссий. При этом требования к БПЛА растут и необходимы системы обеспечивающие отклонения БПЛА от курса не более 5-10 метров на дистанциях более 10 км. В данных условиях одними из перспективных методов автономной навигации становятся методы на основе систем технического зрения [1].

С учетом данных требований наиболее перспективными для изучения являются методы одновременной локализации и построения карты на основе технического зрения, известные как алгоритмы SLAM (simultaneous localization and mapping). Основополагающей работой в области SLAM является исследование Смита и Чизмана по представлению и оценке пространственной неопределенности, проведенное в 1986 году [2]. Представленный метод сейчас обобщен до шести степеней свободы, и позволяет оценить взаимосвязь (положения и ориентации) между объектами, а также оценить неопределенности, связанные с этими взаимосвязями.

Для построения карт местности применяются упрощения на основе топологических карт на основе дескрипторов, фиксирующих топологию среды [3]. Данные технологии хорошо отработаны для автономных наземных транспортных средств в городской среде. Однако, для пересеченной и лесной местности, а также в применении к БПЛА многие алгоритмы работают неустойчиво как для извлечения ключевых точек из изображения, так и для последующего описания дескрипторов для SLAM алгоритма.

Традиционные алгоритмы выделения ключевых точек на изображения, такие как SIFT (Scale-Invariant Feature Transform), SURF (Speeded Up Robust Features) и ORB (Oriented FAST and

Rotated BRIEF), основываются на математических и геометрических принципах. Эти алгоритмы используют различные техники для обнаружения ключевых точек на изображении, которые являются инвариантными к изменениям масштаба, поворота и освещения. Традиционные алгоритмы выделения ключевых точек зарекомендовали себя как надежные инструменты для обработки изображений. Однако, с развитием технологий машинного обучения и нейросетевых подходов появляются новые возможности для повышения точности и устойчивости этих алгоритмов, особенно с учетом необходимости запуска данных алгоритмов на борту БПЛА в условиях ограниченных вычислительных ресурсов.

В качестве базового алгоритма для сравнения подходов к выделению ключевых точек был выбран визуальный SLAM (V-SLAM) [4], использующий изображения, полученные с камер видимого диапазона излучений.

2. Оценка алгоритмов извлечения ключевых точек на изображении

Для сопоставления алгоритмов необходимо определить ключевые характеристики, которыми должен обладать алгоритм извлечения ключевых точек на изображениях, чтобы его можно было успешно адаптировать под специфические требования и условия эксплуатации БПЛА на пересеченной, в том числе лесной, местности. По итогам проведенного анализа были выбраны алгоритмы, обладающие высокими показателями эффективности и широко применяемые в различных системах: SIFT [5], SURF [6], ORB [7], SuperPoint [8] и R2D2 [9]. Они демонстрируют хорошее качество обнаружения ключевых точек, устойчивость к шумам и изменениям в окружающей среде, однако вынуждены балансировать между точностью и способностью быстро обрабатывать данные в реальном времени.

При необходимости использовать алгоритм на борту беспилотника, важно, чтобы он работал в режиме реального времени, особенно если БПЛА выполняет задачи навигации или обнаружения объектов. Соответственно быстрота при извлечении и сопоставлении ключевых точек критически важна для применения алгоритмов на борту БПЛА. Поэтому далее введем метрику для оценки качества алгоритмов и проведем оценку скорости их работы.

Каждый алгоритм извлечения ключевых точек связан с соответствующими алгоритмами описания дескрипторов и последующего их сопоставления, поэтому особый интерес представляет комплексное сравнение данных алгоритмов по времени выполнения и качеству полученных сопоставлений. Для решения этой задачи был подготовлен тестовый набор данных.

2.1 Исходные данные для оценки

Для оценки качества работы алгоритмов был подготовлен набор данных по реальным полетам БПЛА над пересеченной местностью с учетом различных условий: времени суток (день, утро, вечер), времен года, освещения и погодных аномалий (солнечно, переменная облачность, облачно, слабый дождь и др.). Выборка составила более 300 полетов продолжительностью от 10 до 30 минут. Некоторые специфические условия сьемки приведены на рис. 1. Набор данных содержит различные виды подстилающей поверхности от полей и перелесков до лесной местности. Во всех случаях наличие объектов искусственного происхождения минимально (в основном редкие сельские строения и дороги).

Для сьемки использовались БПЛА мультироторного типа (преимущественно квадрокоптеры) с трехосевым подвесом и преимущественно с камерой SIYI ZR10. Полетное задание варьировалось, однако высота составляла порядка 100 метров, скорость в среднем от 5 до 10 м/с, а также был фиксирован ракурс сьемки – камера направлена под углом 30 градусов к вертикали, направления камеры выбирались разными – от совпадающего с курсом полета

БПЛА до направления в сторону (углы 0, 45 и 90 градусов). Частота съемки составила 30 кадров в секунду.

Записи полетов БПЛА были разбиты по кадрам. Каждому кадру соответствовали определенные координаты БПЛА в системе WGS84, а также точные координаты подвеса камеры и параметры сьемки (фокусное расстояние, время электронного затвора и др.). Указанные данные позволили формировать связанные наборы изображений с большим межкадровым смещением. В настоящее время набор данных постоянно пополняется новыми сьемками полетов.

Рис. 1. Примеры различных условий сьемки:
с длинными тенями на закате (слева) и в зимний период (справа).
Fig. 1. Examples of shooting conditions:
with long shadows at sunset (left) and in winter (right).

2.2 Оценка скорости работы алгоритмов

Для оценки скорости работы алгоритмов использовался микрокомпьютер Orange Pi 3 LTS 8Гб с gpu Mali-T720. Все вычисления запускаются на чистой ОС Linux Debian 3.0.8 с установленными зависимостями для выполнения соответствующих алгоритмов. Изображения считываются с встроенной карты памяти.

Записи полетов беспилотника были разбиты по кадрам в случайном порядке из расчета расстояния между крайними кадрами по времени в 1 секунду. Была составлена выборка из 100 000 последовательностей кадров в случайном порядке. Пример пары крайних кадров последовательности приведен на рис. 2.

Рис. 2. Пример двух изображений тестовой выборки (смещение беспилотника в горизонтальной плоскости порядка 10 м, шаг по времени 1 секунда, высота полета 100 м, угол наклона камеры от вертикали 30 градусов, камера направлена по курсу полета).
Fig. 2. An example of two dataset images (horizontal drone displacement - 10 m, time step - 1 second, flight altitude - 100 m, camera tilt angle from the vertical - 30 degrees, camera directed along the flight course)..

Для каждой из 100 тыс. последовательностей кадров производилось извлечение и сопоставление точек для каждого из методов по 10 раз и вычислялось среднее время,

затраченное на исполнение алгоритмов по всей выборке. Сопоставление извлеченных признаков осуществлялось методом k-ближайших соседей из библиотеки FLANN [10-11], а также был рассмотрен нейросетевой метод сопоставления дескрипторов LightGlue для алгоритмов SIFT и SuperPoint. Полученные средние значения времени исполнения алгоритмов на экстракцию ключевых точек и их сопоставление, представлены в табл. 1.

Табл. 1. Результаты сравнения скорости выполнения алгоритмов.
Table 1. Execution time for different algorithms.

Алгоритмы	*SIFT*	*SURF*	*ORB*	*SuperPoint*	*R2D2*	*SIFT и LightGlue*	*SuperPoint и LightGlue*
Время, мс.	237	188	81	101	293	184	54

По итогам сравнения можно сделать вывод, что нейросетевые методы при условии использования соответствующих аппаратных ускорителей показывают наиболее быстрые результаты. Так, наименьшее время работы показала комбинация двух нейросетевых методов – SuperPoint для извлечения ключевых точек и LightGlue для сопоставления дескрипторов. Следующим по скорости выполнения оказался алгоритм ORB, что ожидаемо, так как скорость работы и нетребовательность к вычислительным ресурсам являются главными преимуществами данного традиционного алгоритма. Медленнее всех по итогам тестирования оказались алгоритмы SIFT и нейросеть R2D2, которые требуют больше вычислительных затрат при извлечении и описании ключевых признаков.

Однако, помимо скорости выполнения важно обеспечить качество полученных дескрипторов для последующих операций построения карты и сравнения в рамках SLAM алгоритма. Для оценки качества полученных ключевых точек предлагается подход на основе метрики прослеживаемости ключевых точек между кадрами.

2.3 Оценка прослеживаемости ключевых точек между кадрами

Помимо скорости работы, при полете БПЛА для качественной навигации важно чтобы алгоритм обладал таким свойством, как способность правильно идентифицировать и отслеживать одну и ту же ключевую точку на большом количестве последовательных кадров, несмотря на смещение изображения. Для оценки прослеживаемости ключевых точек между кадрами предлагается использовать следующую метрику:

$$M_n = \frac{1}{n-1}\sum_{i=1}^{n-1}\frac{c_{0,i}}{N_i(1-\alpha i)}$$

где: M_n – усредненная доля правильно сопоставленных ключевых точек на окне из n кадров, n – количество последовательно сопоставляемых кадров, $c_{0,i}$ – количество ключевых точек, успешно сопоставленных между начальным и i-тым кадром, N_i – общее количество выделенных на i-том кадре ключевых точек, α – доля площади поверхности, на которую различается i-тый и начальный кадры без учета рельефа местности.

Данная метрика направлена на оценку продолжительности сопоставления ключевых точек на последовательности кадров, что позволяет более точно оценить эффективность алгоритмов в динамичных условиях пересеченной местности для различных условий.

Для коэффициента α использованы упрощения – так как полеты выполняются в основном в равнинной местности, либо с небольшими перепадами рельефа, то поверхность можно представить в виде плоскости и вычислить разницу площадей между кадрами в упрощенной постановке.

Например, рассмотрим равномерное прямолинейное движение беспилотника над равнинной местностью. Если камера направлена вдоль оси Z, а оси X и Y определяют плоскость изображения, то с помощью библиотеки *geopy* и метаданных тестовых полетов можно

рассчитать, сколько пролетает БПЛА, для того чтобы заснять полностью новый кадр. а в общем случае получить направление камеры в новом положении. Коэффициентами дисторсии в нашем случае можно пренебречь, т.к. изображения сохранены уже после соответствующих преобразований.

Предположим, что скорость БПЛА составляет 1 м/с, высота 100 метров, тогда из данных телеметрии определяется смещение между двумя соседними кадрами, например 0.96 м, изменение площадей получим через проецирование изображения камеры на поверхность Земли. Исходя из данных вычислений α = 0.0018. То есть между двумя последовательными кадрами, совпадает примерно 98.2% площади изображенной на кадре поверхности Земли. Или по-другому, площадь, покрываемая на текущем кадре, отличается от площади на предыдущем снимке на 1.8%.

Для сравнительного анализа традиционных и нейросетевых подходов были взяты серии кадров по ограничению на коэффициент α не более 30% различия площади. На данных сериях кадоров были оценены алгоритмы выделения ключевых точек по метрике M_n на собранных изображениях пересеченной местности. Были рассмотрены приведенные выше методы.

Кроме ограничений, связанных с пересечением площадей реальной поверхности Земли, некоторые алгоритмы имеют настраиваемые параметры, одним из наиболее влияющих на качество последующей работы SLAM алгоритмов является ограничение по количеству извлекаемых точек на изображении. Поэтому помимо измерения метрик на наборе данных необходимо было выбрать оптимальные параметры для каждого из алгоритмов. Для этого для рассматриваемых методов при различных ограничениях максимального количества извлекаемых точек подсчитывалась предложенная метрика M_5. На 200 изображениях для каждого окна из 5 кадров вычислим значение метрики, а затем найдём среднее по всем вычислениям. Получившиеся результаты представлены на рис. 3.

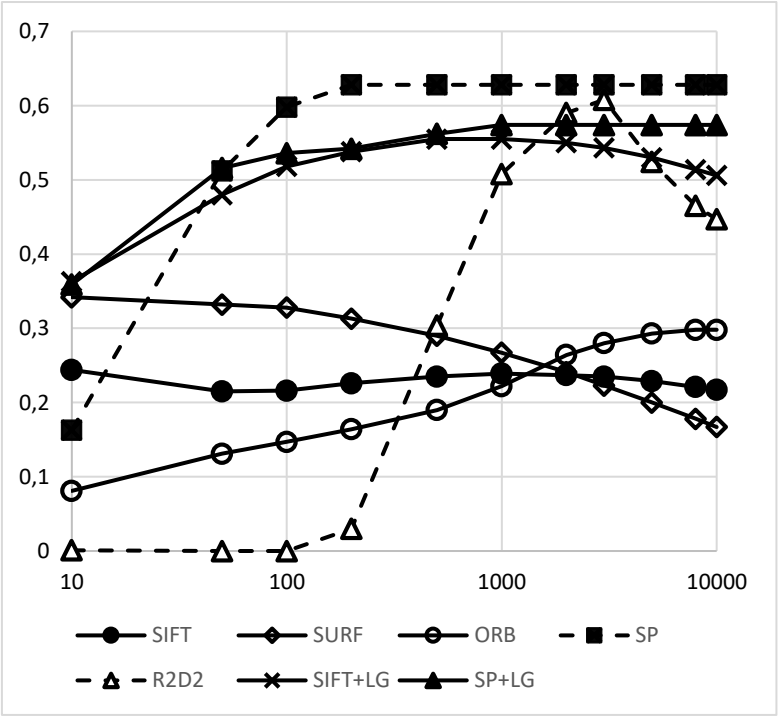


Рис. 3 Значение метрики M5 в зависимости от ограничения числа извлекаемых ключевых точек.
Fig. 3. M5 value depending of the extracted key points number limit.

На рис. 3 можно заметить, что самыми высокими значениями метрики обладает нейросетевая модель SuperPoint. Хотя моделью и извлекается не так много точек (табл. 2), но они являются устойчивыми и продолжительно сопоставляются на последовательности кадров. Если вместо метода k ближайших соседей из библиотеки FLANN использовать для сопоставления извлеченных признаков нейросеть LightGlue, то эксперимент тоже показывает достаточно хорошие результаты, при этом после фильтрации остается больше хорошо отслеживаемых ключевых точек, по сравнению с SuperPoint + KNN (табл. 2). Алгоритм SIFT показывает несколько худший результат в сочетании с LightGlue, а нейросеть R2D2 показывает высокое значение на ограничении порядка 3000 извлекаемых ключевых точек.

Сравнение показывает, что традиционные способы извлечения и описания ключевых точек значительно проигрывают нейросетевым подходам в случае, когда важно извлекать устойчивые признаки, которые будут продолжительно отслеживаться на последовательности кадров одной динамической сцены. По итогам сравнения различных методов извлечения ключевых точек и сопоставления полученных дескрипторов для реализации на борту БПЛА в качестве компонента SLAM алгоритма можно рекомендовать комбинацию двух нейросетевых подходов *SuperPoint и LightGlue*. Кроме этого, указанные решения могут быть дообучены для обеспечения большей точности извлечения ключевых точек и сопоставления дескрипторов. Также на основе проделанной работы была оценена скорость выполнения алгоритмов, что поможет в дальнейшем выбрать наиболее подходящий бортовой вычислитель для системы автономной навигации БПЛА.

Табл. 2. Реальное количество извлекаемых точек при различном значении ограничения.
Table 2. Actual number of points extracted for different limit values.

N	***SIFT***	***SURF***	***ORB***	***SuperPoint***	***R2D2***	***SIFT + LightGlue***	***SuperPoint + LightGlue***
10	10	10	10	10	10	7	10
50	50	50	50	49	50	34	50
100	100	100	100	88	100	70	100
200	200	200	200	128	200	142	200
500	500	500	500	129	500	362	495
1000	1000	1000	1000	129	1000	734	733
2000	2000	2000	2000	129	2000	1485	734
3000	3000	3000	3000	129	3000	2247	734
5000	5000	5000	5000	129	5000	3787	734
8000	8000	8000	8000	129	8000	6126	734
10000	10000	10000	10000	129	10000	7544	734

Список литературы / References

[1]. Али Б., Садеков Р.Н., Цодокова В.В. Алгоритмы навигации беспилотных летательных аппаратов с использованием систем технического зрения. Гироскопия и навигация. Том 30. №4 (119), 2022, стр.87-105. DOI 10.17285/0869-7035.00105.

[2]. Smith R.C., Cheeseman P. On the Representation and Estimation of Spatial Uncertainty. The International Journal of Robotics Research. 5 (4): 56–68. 1986. DOI: 10.1177/027836498600500404.

[3]. Cummins M., Newman P. FAB-MAP: Probabilistic localization and mapping in the space of appearance. The International Journal of Robotics Research. 27 (6): 647–665, 2008. DOI:10.1177/0278364908090961.

[4]. Yalan Chen, Yimin Zhou, Qin Lv, Kranthi Kumar Deveerasetty A Review of V-SLAM. Conference: 2018 IEEE International Conference on Information and Automation (ICIA). 2018. DOI: 10.1109/ICInfA.2018.8812387.

[5]. Lowe D.G. Distinctive Image Features from Scale-Invariant Key Points. International Journal of Computer Vision, 2004, 60, 91-110. DOI 10.1023/B:VISI.0000029664.99615.94.
[6]. Bay H., Tuytelaars T., Van Gool, L. SURF: Speeded Up Robust Features. In: Leonardis, A., Bischof, H., Pinz, A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951. Springer, Berlin, Heidelberg. DOI 10.1007/11744023_32.
[7]. Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary Bradski, Gary Bradski ORB: an efficient alternative to SIFT or SURF. Conference: IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011. DOI: 10.1109/ICCV.2011.6126544.
[8]. Daniel DeTone, Tomasz Malisiewicz, Andrew Rabinovich SuperPoint: Self-Supervised Interest Point Detection and Description. Conference: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). 2018. DOI: 10.1109/CVPRW.2018.00060.
[9]. Jerome Revaud, Philippe Weinzaepfel, César De Souza, Martin Humenberger. R2D2: Repeatable and Reliable Detector and Descriptor. Conference on Neural Information Processing Systems (NeurIPS), Vancouver, Canada, 8-14 December, 2019. DOI 10.48550/arXiv.1906.06195.
[10]. Marius Muja and David G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration", in International Conference on Computer Vision Theory and Applications (VISAPP'09), Vol.1, 2009.
[11]. Библиотека FLANN [Электронный ресурс] // GitHub. – 2009. – Режим доступа: https://github.com/flann-lib/flann (дата обращения: 04.05.2025).

Информация об авторах / Information about authors

Петр Александрович УХОВ – кандидат технических наук, доцент, заместитель начальника управления «IT-Центр» с 2020 года. Сфера научных интересов: машинное обучение, компьютерное зрение, программное обеспечение беспилотных авиационных систем, предиктивная аналитика, системы управления и навигации летательных аппаратов.

Peter Alexandrovich UKHOV – Cand. Sci. (Tech.), Associate professor, Deputy Head of «IT-Centre» MAI since 2020. Research interests: machine learning, computer vision, unmanned aircraft systems software, predictive analytics, aircraft control and navigation systems.

Мария Борисовна БУЛАКИНА – кандидат технических наук, доцент, начальник управления «IT-центр» Московского авиационного института. Область научных интересов – технология разработки программных систем, системный дизайн, управление IT-проектами.

Maria Borisovna BULAKINA – Cand. Sci. (Tech.), Associate Professor, Head of the IT Center Department at the Moscow Aviation Institute. Her research interests include software systems development technology, system design, and IT project management.

Сергей Сергеевич КРЫЛОВ – кандидат физико-математических наук, доцент, заведующий кафедрой «Вычислительная математика и программирование» Московского авиационного института. Область научных интересов – технология разработки программных систем, системы геометрического моделирования.

Sergey Sergeevich KRYLOV – Cand. Sci. (Phys.-Math.), Associate Professor, Head of the Department of Computational Mathematics and Programming at the Moscow Aviation Institute. His research interests include software systems development technology and geometric modeling systems.

DOI: 10.15514/ISPRAS-2025-37(6)-25

Применение нейронных сетей семейства YOLO для обнаружения полезных сигналов на вихретоковых дефектограммах рельсов

А.Н. Гладков, ORCID: 0009-0007-0211-5660 <a.gladkov@uniyar.ac.ru>
Л.Ю. Быстров, ORCID: 0000-0002-0610-5466 <l.bystrov@uniyar.ac.ru>
Е.В. Кузьмин, ORCID: 0000-0003-0500-306X <kuzmin@uniyar.ac.ru>

Ярославский государственный университет им. П.Г. Демидова,
Россия, 150003, Ярославль, ул. Советская, д. 14.

Аннотация. Повышение уровня безопасности железнодорожного движения напрямую связано с необходимостью оперативного обнаружения структурных аномалий элементов рельсового пути. Данная задача реализуется посредством регулярных проверок состояния рельсов с применением методов неразрушающего контроля. Среди современных технологий, используемых для этой цели, выделяется вихретоковая дефектоскопия. Дефектоскоп формирует многоканальный дискретный сигнал, который называется дефектограммой. Дефектограммы нуждаются в анализе, то есть в выявлении полезных сигналов, указывающих на дефект или конструктивные элементы рельса. В работе рассматривается применение детектирующих свёрточных нейронных сетей семейства YOLO (You Only Look Once) для автоматического обнаружения полезных сигналов рельсов на вихретоковых дефектограммах рельсов. Цель исследования – оценить эффективность различных способов преобразования многоканального сигнала в двумерные изображения, совместимые с YOLO. Исследованы четыре метода преобразования: пороговое, основанное на сравнении амплитуд с пороговым уровнем шума, оконное преобразование Фурье, непрерывное вейвлет-преобразование и преобразование Гильберта-Хуанга. Набор данных для обучения состоит из фрагментов дефектограмм по 50 тыс. отсчётов с полезными сигналами трёх классов (болтовые стыки, электроконтактные и алюминотермитные сварки). Данные разделены на обучающую, валидационную и тестовую выборки. Обученные на этих данных модели YOLO для всех рассмотренных методов преобразования продемонстрировали высокие показатели сбалансированной средней точности mAP. Наилучшие показатели были достигнуты при использовании непрерывного вейвлет-преобразования, в то время как пороговое преобразование оказалось наименее вычислительно затратным. Оконное преобразование Фурье позволило достичь лучшего баланса между точностью и полнотой обнаружения полезных сигналов. Результаты исследования подтверждают потенциал использования сетей YOLO для анализа вихретоковых дефектограмм и сигналов в целом.

Ключевые слова: свёрточные нейронные сети; сети YOLO; вихретоковая дефектоскопия; оконное преобразование Фурье; непрерывное вейвлет-преобразование; преобразование Гильберта-Хуанга.

Для цитирования: Гладков А.Н., Быстров Л.Ю., Кузьмин Е. В. Применение нейронных сетей семейства YOLO для обнаружения полезных сигналов на вихретоковых дефектограммах рельсов. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 131–150. DOI: 10.15514/ISPRAS–2025–37(6)–25.

Благодарности: Исследование выполнено в рамках инициативной НИР ЯрГУ им. П.Г. Демидова № VIP-016.

Application of YOLO Family Neural Networks for Useful Signals Detection on Eddy Current Rail Defectograms

A.N. Gladkov, ORCID: 0009-0007-0211-5660 <a.gladkov@uniyar.ac.ru>
L.Y. Bystrov, ORCID: 0000-0002-0610-5466 <l.bystrov@uniyar.ac.ru>
E.V. Kuzmin, ORCID: 0000-0003-0500-306X <kuzmin@uniyar.ac.ru>
P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl 150003, Russia.

Abstract. Improving the level of safety of railway traffic is directly related to the need for prompt detection of structural anomalies of track elements. This task is implemented through regular inspections using non-destructive testing methods. Among the modern technologies used for this purpose, eddy current flaw detection stands out. The flaw detector generates a multi-channel discrete signal, which is called a defectogram. Defectograms require analysis, that is, the identification of useful signals from a defect or structural elements of the rail. This paper investigates the use of YOLO (You Only Look Once) family convolutional neural networks for automated detection of useful signals in eddy current rail defectograms. The main objective was to evaluate the effectiveness of different transformations of multichannel time-series data into two-dimensional images suitable for YOLO processing, and to explore the trade-off between detection accuracy and computational cost. Four transformation methods are examined: Threshold Transform, based on amplitude comparisons against a twice threshold noise level, Short-Time Fourier Transform, Continuous Wavelet Transform and Hilbert–Huang Transform. The dataset comprises defectogram fragments of 50 thousand counts with annotated useful signals from three classes (flash butt welds, aluminothermic welds, and bolt joints), split into training, validation, and test sets. YOLO models trained on this data achieved high mean Average Precision scores in useful signals detection for all considered transformation methods. Continuous Wavelet Transform yielded the best scores while the Threshold Transform proved to be the least computationally expensive. Short-Time Fourier Transform method offered the best balance between precision and recall. Hilbert–Huang Transform showed slightly lower effectiveness. These results demonstrate the suitability of YOLO networks for eddy current defectogram analysis and useful signals detection in general.

Keywords: convolutional neuron networks; YOLO neuron networks; eddy current testing; short-time Fourier transform; continuous wavelet transform; Hilbert–Huang transform.

For citation: Gladkov A.N., Bystrov L.Y., Kuzmin E.V. Application of YOLO Family Neural Networks for Useful Signals Detection on Eddy Current Rail Defectograms. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 131-150 (in Russian). DOI: 10.15514/ISPRAS-2025-37(6)-25.

Acknowledgements. This work was supported by P.G. Demidov Yaroslavl State University Project № VIP-016.

1. Введение

Вихретоковая дефектоскопия – это один из современных методов неразрушающего контроля железнодорожного полотна, основанный на взаимодействии вихревых токов с материалом рельса. Он применяется для выявления дефектов в электропроводящих материалах без повреждения объекта. Дефектоскоп формирует сигналы для каждого миллиметра рельса с помощью набора физических датчиков. Использование нескольких датчиков придаёт сигналам пространственную структуру. Количество датчиков определяет число каналов для каждого сигнала. Полученный таким образом набор данных называется дефектограммой. Дефектограммы отражают информацию о расположении конструктивных элементов и состоянии поверхности катания рельсов (рис. 1). Анализ дефектограмм заключается в поиске на ней сигналов от дефектов и конструктивных на фоне рельсового шума и помех разного рода. Системы автоматического анализа дефектограмм позволяют своевременно обнаружить и устранить дефекты рельсов, что прямым образом влияет на безопасность движения железнодорожного транспорта.

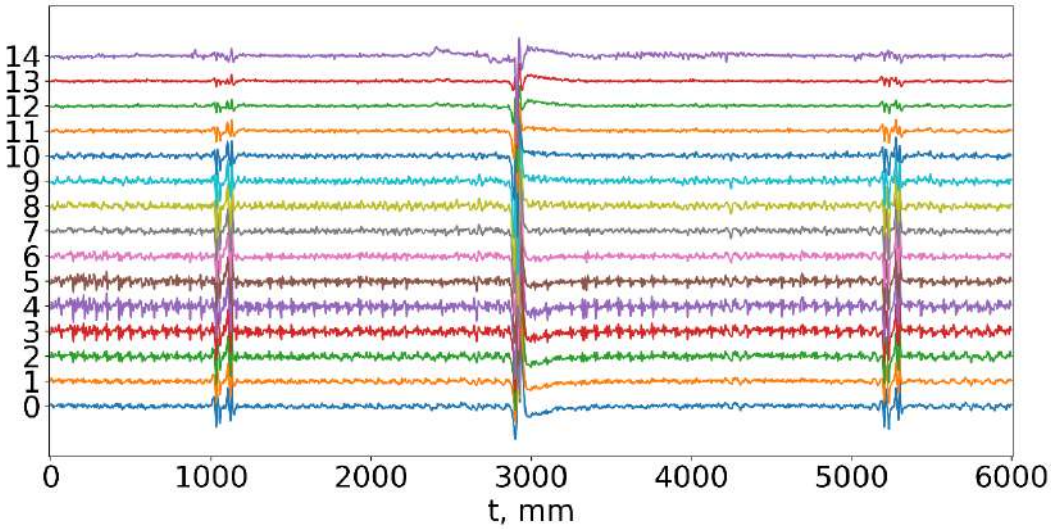


Рис. 1. Пример вихретоковой дефектограммы рельсов.
Fig. 1. The Example of Eddy Current Rail Defectogram.

В данной работе рассматриваются дефектограммы, сформированные 14-разрядным 15-канальным вихретоковым дефектоскопом. Значения амплитуд сигналов – целые числа от -8192 до 8191. Ввиду большой протяжённости дефектограммы нарезают на фрагменты. Длина одного фрагмента дефектограммы – 50 тыс. отсчётов. Далее под дефектограммой будут подразумеваться её фрагменты описанного размера. Таким образом, дефектограммы можно рассматривать как 15-канальный дискретный сигнал $\bar{x}(j)$:

$$\bar{x}(j) = \big(x_0(j), x_1(j), \dots, x_{14}(j)\big), \qquad j \in \overline{0{,}49999}.$$

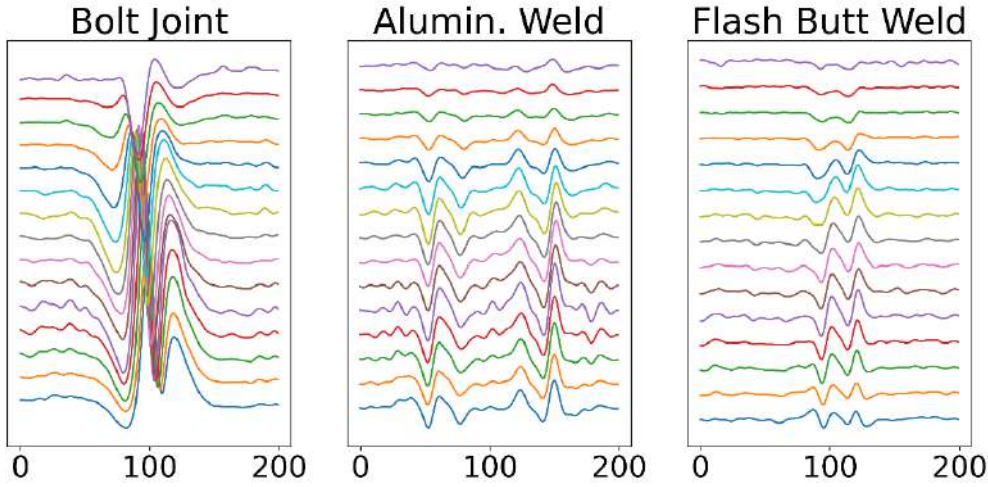


Рис. 2. Примеры сигналов коротких конструктивных элементов на вихретоковых дефектограммах рельсов.
Fig. 2. The Examples of Short Structural Element Signals on Eddy Current Rail Defectogram.

На вихретоковых дефектограммах можно выделить сигналы от коротких и протяжённых конструктивных элементов, помех оборудования и рельсовых дефектов. Среди полезных

сигналов дефектограммы в данной статье рассматриваются сигналы, формирующие образы коротких конструктивов: болтовых стыков (Bolt Joint), электроконтактных (Flash Butt Weld) и алюминотермитных сварок (Aluminothermic Weld). Эти образы представлены на рис. 2. Стандартный подход к обнаружению полезных сигналов в дефектограмме основан на вычислении порогового уровня рельсового шума [1]. Полезные сигналы выделяются на фоне шума высокими значениями амплитуд, поэтому естественной идеей их обнаружения является сравнение амплитуд сигналов со среднеквадратическим отклонением рельсового шума. Перечислим проблемы такого подхода:

- Пороговый уровень шума может быть завышен из-за присутствующих в данных помех [2].
- Не все полезные сигналы могут выходить за пороговый уровень. Возникают трудности с точностью выделения границ полезных сигналов.
- Полезные сигналы могут быть записаны с более низкой чувствительностью по сравнению с рельсовым шумом и не выходить за пределы порогового уровня.

Таким образом, существует потребность в поиске новых методов обнаружения полезных сигналов на вихретоковых дефектограммах, которые могли бы учитывать форму и сложные свойства сигналов, а не только их высокоамплитудность.

В последние годы активно развивается подход представления одномерных сигналов в виде двумерных изображений для последующей обработки свёрточными нейронными сетями (Convolutional Neuron Network, CNN) [3]. Такие методы позволяют использовать проверенные архитектуры компьютерного зрения для задач анализа сигналов. В задаче обнаружения объектов нейронные сети могут выступать в качестве основного инструмента, в этом случае вся логика решения реализуется в архитектуре сети напрямую и сеть носит название детектирующей. Наиболее популярным семейством детектирующих свёрточных нейронных сетей являются сети YOLO (You Only Look Once), демонстрирующие впечатляющие результаты в области анализа изображений. Модели YOLO обеспечивают баланс между точностью и скоростью обнаружения объектов в реальном времени [4]. В отличие от двухэтапных детекторов, таких как R-CNN, разделяющими задачу формирования ограничивающих рамок (bounding boxes) в одном модуле и их отбор в другом, YOLO рассматривает задачу обнаружения объектов как проблему регрессии, предсказывая ограничивающие рамки и вероятности классов объектов непосредственно из входного изображения за один проход через нейронную сеть. Изначальная архитектура YOLO состояла из свёрточных слоёв и слоёв макс-пулинга (max-pooling), но с течением времени она эволюционировала в более модульную структуру. Каждая новая версия YOLO вносила улучшения в архитектуру, функцию потерь и методы обучения, что приводило к повышению точности обнаружения, особенно для объектов малого размера, и снижению вычислительных затрат [5]. Высокая скорость работы и эффективность YOLO позволяют использовать эти сети для обработки потенциально больших объёмов данных в реальном времени.

В данной работе исследуются методы применения сетей YOLO в решении задачи обнаружения полезных сигналов на вихретоковых дефектограммах. Методы отличаются друг от друга способом преобразования 15-канального сигнала $\bar{x}(j)$ в двумерное изображение $F(x; y)$. Рассматриваются 4 вида преобразований: оконное преобразование Фурье, непрерывное вейвлет-преобразование, преобразование Гильберта-Хуанга и пороговое преобразование. Пороговое преобразование является новым методом и вводится впервые. Все виды рассмотренных методов успешно интегрируются с YOLO и показывают высокое качество обнаружения полезных сигналов.

2. Обзор литературы

Существует большое количество техник преобразования сигналов в изображения. Среди них можно выделить две основные группы: преобразования, основанные на частотно-временном анализе, и методы векторного кодирования изображений. К первой категории относятся спектрограммы (Short-time Fourier Transform, STFT), скалограммы (Continuous Wavelet Transform, CWT), спектры Гильберта-Хуанга (Hilbert-Huang Transform, HHT), распределения Вигнера-Вилля (Wigner-Ville Distribution, WVD). Вторая категория представлена грамиановыми угловыми полями (Gramian Angular Field, GAF), рекуррентными изображениями (Recurrence Plot, RP), разностными полями Маркова (Markov Transition Field, MTF) и полями разности мотивов (Motif Difference Field, MDF). Отдельно от этих двух больших групп отстоят изображения гомологической устойчивости (Persistence Image, PI). Все эти преобразования визуально представлены на рис. 3.

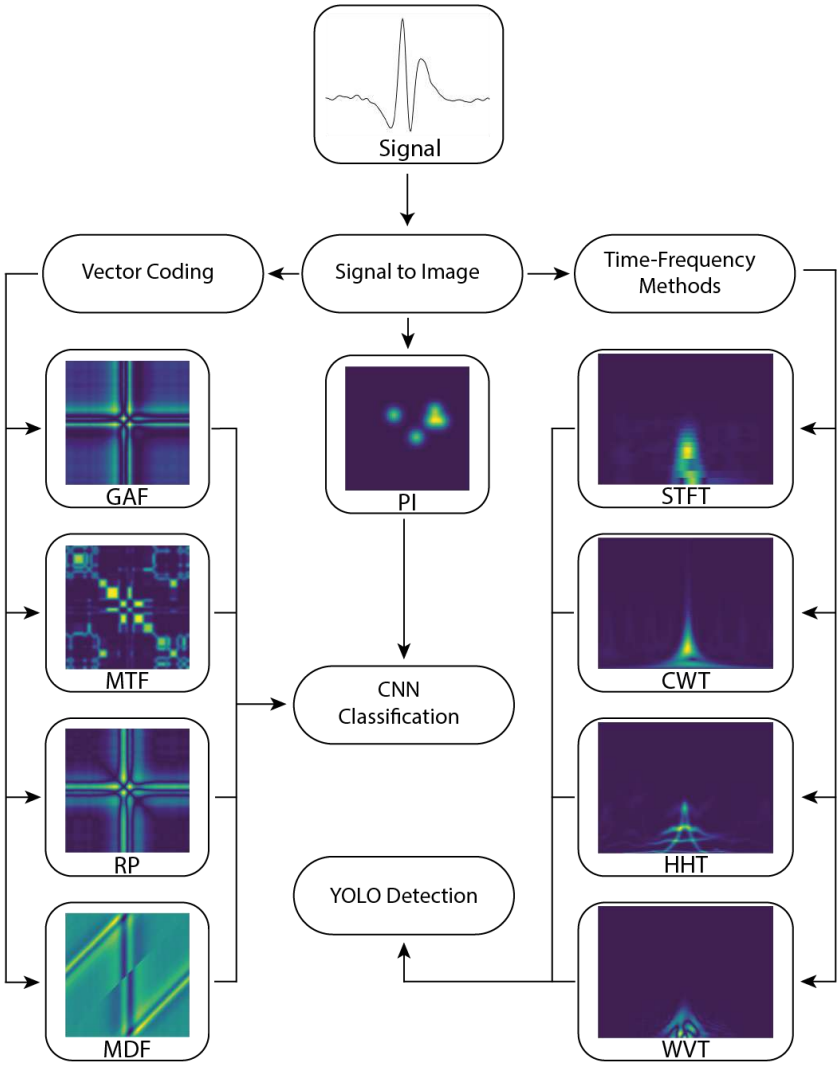


Рис. 3. Основные способы преобразования сигналов в изображения.
Fig. 3. The Main Ways to Convert Signals into Images.

Оконное преобразование Фурье (STFT) позволяет анализировать гармонический спектр в зависимости от времени. В дискретном случае оно описывается функцией

$$F(t,\omega) = \frac{1}{N}\sum_{j=0}^{N-1} x(j)\cdot W(j-t)\cdot e^{-\frac{i\omega 2\pi}{N}},$$

где $x(j)$ – дискретный сигнал, $j \in \overline{0,N-1}$, $W(t)$ – оконная функция, $\omega \in \overline{0,N-1}$, $i = \sqrt{-1}$.

Этот метод широко используется в распознавании аудио- и вибросигналов [6]. Получающиеся с помощью него изображения называются спектрограммами. Спектрограмма отражает как временную, так и частотную составляющую сигнала, что может быть полезно при поиске полезных сигналов. Спектрограммы являются наиболее популярными двумерными представлениями сигналов, активно применяющимися на практике. Так в работах [7] и [8] для вибрационных сигналов, полученных от электродвигателей и вращающихся машин, были построены спектрограммы, на которых обучена свёрточная нейронная сеть для классификации. Такой подход позволил значительно усовершенствовать системы диагностики неисправностей. Аналогичный подход применялся в статьях [9] и [10] и позволил достичь высокого качества обнаружения дефектов в свёрлах и дорожном покрытии. Авторы [11] применили сеть YOLOv10 вместе с STFT в задаче обнаружения и классификации радиосигналов и показали важность правильного выбора оконной функции для качества обнаружения. Успех применения спектрограмм в данных задачах обусловлен тем, что вибрации вращающихся систем, звуковые колебания и радиосигналы характеризуются выраженными гармоническими составляющими. Если же сигналы являются нестационарными и обладают сложной гармонической структурой, спектрограмма может не отражать их особенностей. В этом случае применяют непрерывное вейвлет-преобразование [12].

Главным отличием вейвлет-преобразования от преобразования Фурье является использование в разложении сигналов в качестве базиса не бесконечных функций, а ограниченных, локализованных по времени и по частоте. Это позволяет выявлять в сигналах резкие и быстро затухающие скачки амплитуд. Непрерывное вейвлет-преобразование $F(a,b)$ – это свёртка исходного ряда $x(j)$ с материнской вейвлет-функцией $\psi(j)$:

$$F(a,b) = \frac{1}{\sqrt{a}}\sum_{j=0}^{N-1} x(j)\cdot \psi\left(\frac{j-b}{a}\right),$$

где a – параметр масштаба, b – смещение, $a > 0$, $b \in \overline{0,N-1}$.

Получаемые на основе CWT изображения называются скалограммами (от англ. $scale$ – масштаб). Скалограммы использовались в [13] для обработки вибрационных сигналов с целью повышения производительности моделей на основе CNN при обнаружении неисправностей в винтовых редукторах, благодаря чему была достигнута высокая точность классификации (более 99%). Авторы [14] применяли CWT в качестве промежуточного слоя в архитектурах LSTM и CNN для классификации аномалий в вентиляторных системах. В работе [15] проводится сравнительный анализ непрерывных вейвлет-преобразований с разными материнскими вейвлет-функциями (WT-Amor, WT-Bump, WT-Morse) в задаче диагностики неисправностей электродвигателей с использованием свёрточных нейронных сетей. Самая высокая доля правильных ответов (accuracy) была достигнута при использовании вейвлетов семейства Морзе. Авторы [16] решали задачу обнаружения утечек в трубопроводе. Они подавали на вход свёрточной сети двухканальное изображение, где один канал представлял собой спектограмму, а второй – скалограмму сигналов акустической эмиссии. Такое простое сочетание методов позволило значительно улучшить качество обнаружения утечек.

Итак, непрерывное вейвлет-преобразование является универсальным инструментом преобразования различных типов сигналов в информативные для CNN изображения. Главное

ограничение CWT – высокая трудоёмкость по времени и по памяти при генерации матриц [17] (прогнозируется квадратичная сложность по размеру сигнала).

Отдельно остановимся на статье [18], авторы которой решали задачу обнаружения полезных сигналов в вихретоковых дефектограммах рельсов. Изображения в статье формируются с использованием непрерывного вейвлет-преобразования Морле. Авторы заявляют, что им удалось для сформированных изображений обучить свёрточную нейронную сеть и классифицировать с 98% точностью полезные сигналы трёх видов: скваты (вид дефекта), сварки и болтовые стыки. Однако результаты исследования нуждаются в тщательной проверке. Сомнения в полученных результатах вызывает использование авторами одноканального дефектоскопа, который, очевидно, даёт ограниченное представление о состоянии поверхности катания рельсов. Более того, наличие всего лишь одного канала затрудняет формирование образов конструктивных элементов рельсов (сварок и стыков рельсов). Полезные сигналы обнаруживаются с помощью сравнения амплитуд с постоянным пороговым уровнем шума при том, что уровень шума рельсов, как правило, является динамической величиной (зависит от настроек оборудования и типов рельсов) и для разных участков рельсового пути может сильно отличаться. Таким образом, достоверность результатов исследования [18] вызывает сомнения, результаты работы нуждаются в апробации.

Преобразование Гильберта-Хуанга (HHT) – это ещё один метод анализа данных, предназначенный для нелинейных и нестационарных сигналов, то есть сигналов, частотные характеристики которых меняются со временем. HHT состоит из двух этапов: эмпирическая модовая декомпозиция и преобразование Гильберта. На первом этапе сигнал $x(t)$ разлагается на набор простых колебательных компонент, называемых внутренними модовыми функциями (Intrinsic Mode Functions, IMF). IMF являются монохроматическими функциями, то есть локально выглядят как синусоиды с медленно меняющимися параметрами. В результате эмпирической модовой декомпозиции ряд раскладывается на n модовых функций c_j и остаток r:

$$x(t) = \sum_{j=1}^{n} c_j(t) + r(t).$$

Далее, к каждой IMF применяется преобразование Гильберта

$$H\big[c_j(t)\big] = \frac{1}{\pi}\, P.V. \int_{-\infty}^{+\infty} \frac{c_j(\tau)}{t-\tau} d\tau,$$

где $P.V.$ означает главное значение интеграла по Коши (Cauchy Principal Value).

Сигнал вида $h_j(t) = c_j(t) + iH\big[c_j(t)\big]$ называется аналитическим. Представление аналитического сигнала в полярной форме позволяет перейти к спектру Гильберта.

$$h_j(t) = A_j(t)e^{i\Theta_j(t)}, A_j(t) = \sqrt{\Big(c_j(t)\Big)^2 + \big(H\big[c_j(t)\big]\big)^2}, \Theta_j(t) = \arctan\frac{H\big[c_j(t)\big]}{c_j(t)}.$$

Мгновенная частота в спектре находится как производная фазы по времени

$$\omega_j(t) = \frac{1}{2\pi}\frac{d\Theta_j(t)}{dt}.$$

Наконец, набор троек $\Big(t, \omega_j(t), A_j(t)\Big)$ образует спектр Гильберта для всех IMF j и моментов времени t. Такое представление даёт богатую информацию о динамике нестационарных процессов. При этом преобразование Гильберта-Хуанга, как и вейвлет-преобразование, может быть вычислительно трудоёмким. Другой недостаток HHT связан с тем, что эмпирическая модовая декомпозиция может иногда приводить к смешению мод, когда различные частотные компоненты смешиваются в одной и той же IMF. В этом случае спектр Гильберта становится тяжело интерпретируемым.

В статье [19] решается задача диагностики неисправностей подшипников в реактивных двигателях с переключаемым сопротивлением. Для этого по вибрационным сигналам строится спектр Гильберта, на котором с помощью двухэтапного детектора R-CNN выполняется поиск ограничивающих рамок. Двухэтапные детекторы значительно уступают в качестве одноэтапным, таким как YOLO. Авторы заявляют о высокой доле правильных ответов, но считают её на обучающей, а не тестовой выборке. В связи с тем, что сам метод R-CNN не доказал свою эффективность [20], возникают сомнения насчёт того, что модель сможет показать такие же впечатляющие результаты обнаружения объектов на новых данных.

Авторы [21] представили автоматическую систему для оценки качества бетона методом ударно-эхового контроля. В ней сигналы преобразовывались в изображения с помощью распределения Вигнера-Вилля (WVD). Далее, в отличие от работ, рассмотренных ранее, свёрточные сети использовались напрямую для обнаружения объектов. Авторы обучили разные версии YOLO и с помощью них обнаружили трещины, поверхностные волны и вибрации в бетоне. При этом вместо классических метрик точности (Precision), полноты (Recall) и средней точности (mAP) они использовали свою метрику Detection rate, равную отношению числа найденных отметок к числу правильных отметок. Интерпретация данной метрики затруднительна. Если сеть найдёт слишком много лишних отметок, то Detection rate превысит единицу. Таким образом, используемая авторами метрика не позволяет адекватно оценить качество работы сети и, соответственно, подвергает сомнению результаты исследования.

Вторая группа преобразований сигналов в изображения основана на векторном кодировании. Ввиду того, что получаемые в данной группе изображения сложно использовать вместе с YOLO, им уделено меньше внимания.

Грамиановы угловые поля (GAF) и разностные поля Маркова (MTF) преобразуют сигнал в матрицу попарных отношений. Это позволяет получать информацию о корреляции всех пар амплитуд во времени [22]. Как правило, GAF/MTF применяются там, где важна общая структура и глобальные временные закономерности. Кратковременные сигналы могут быть размыты или менее заметны. С некоторыми проблемами GAF и MTF справляются поля разностей мотивов (MDF), которые обладают простотой реализации и отлавливают больше значимых признаков сигнала [23]. Известно, что эти методы требуют значительных вычислительных ресурсов (для сигнала размера N создаётся матрица N×N) и порождают изображения большого размера, не пригодного для YOLO. Если исходный сигнал разбить на множество небольших фрагментов и для них выполнить GAF/MTF/MDF, то полученные изображения будут отлично сочетаться с моделями классификации (одно изображение — один класс), но не подойдут для задач, где нужно обнаруживать несколько объектов [24].

Рекуррентные изображения (RP) визуализируют моменты времени, когда траектория в фазовом пространстве возвращается в ранее посещённое состояние. Они создаются путём вычисления матрицы расстояний между всеми парами точек в реконструированном фазовом пространстве и последующей пороговой обработкой этой матрицы для выявления рекуррентности [25]. Рекуррентные изображения имеют все те же проблемы, что и GAF/MTF/MDF: высокая трудоёмкость и большой размер изображений. Кроме того, они выделяют только повторяющиеся закономерности и не кодирует пространственное и временное расположение сигналов, что имеет принципиальное значение для обнаружения полезных сигналов вихретоковых дефектограмм.

Ещё один способ создания изображений из сигналов основан на вычислении устойчивых гомологий. Устойчивые гомологии – это объект из топологического анализа данных, который извлекает информацию о «форме» данных, идентифицируя и количественно определяя топологические признаки, такие как компоненты связности и дыры, которые сохраняются при различных масштабах (разрешениях) данных – фильтрациях. Изображения гомологической устойчивости (PI) – это способ представления выходных данных

топологического анализа (диаграмм устойчивости) в виде изображений с помощью свёртки диаграмм с гауссовским ядром [26]. Они часто используются в машинном обучении [27]. PI отображают время жизни топологических характеристик сигналов, но по ним нельзя восстановить временную локализацию сигналов. Таким образом, изображения устойчивых гомологий можно использовать для классификации сигналов, но не для их обнаружения, то есть сети YOLO к PI не применимы.

Итак, лучше всего для интеграции с YOLO подходят преобразования сигналов в изображения, основанные на частотно-временном анализе. С одной стороны, они сохраняют временную локализацию сигналов, с другой – кроме пространственных признаков позволяют анализировать частотные характеристики. Изображения на основе GAF, MTF, RP и PI могут использоваться вместе с нейросетевыми классификаторами, но не пригодны для детектирующих моделей.

3. Методология

В данном разделе рассматриваются методы преобразования изображений, которые впоследствии интегрируется с YOLO. Три из четырёх методов основаны на частотно-временном анализе и уже были частично рассмотрены в предыдущем разделе: оконное преобразование Фурье, непрерывное вейвлет-преобразование и преобразование Гильберта-Хуанга. Предложен также авторский вариант преобразования, названный пороговым преобразованием (Threshold Transform, TT).

Требуется не просто преобразовать дефектограмму в изображение, но в изображение приемлемого для YOLO размера. Сети YOLO на вход принимают изображения размером 640×640 в формате RGB. Поэтому полученные из дефектограммы изображения нарезаются на несколько фрагментов длиной 640 пикселей каждый. После данной операции фрагментированные изображения выстраиваются вертикально друг под другом и дополнительно разделяются зелёными полосами для наглядного представления границ фрагментации. На основе каждого из преобразований строятся 2 матрицы, соответствующие красному и синему цвету, а зелёный цвет используется для отображения полос. Таким образом достигается размер и формат изображения, воспринимаемые YOLO.

3.1. Пороговое преобразование

Данное преобразование не основано на частотно-временном анализе и вводится впервые. Будем непосредственно сопоставлять каждой амплитуде сигнала пиксель с определённой цветовой интенсивностью. Такой простой подход позволяет значительно сократить вычислительные ресурсы, которые требуются для создания изображений преобразованию Фурье и более трудоёмким алгоритмам, таким как непрерывное вейвлет-преобразование и преобразование Гильберта-Хуанга. На основе дефектограммы $\bar{x}(j)$ построим 2 матрицы $F_r(i,j)$ и $F_b(i,j)$. Матрица $F_r(i,j)$ хранит информацию об интенсивностях красного цвета каждого пикселя и соответствует положительным амплитудам сигналов. Матрица $F_b(i,j)$ хранит информацию об интенсивностях синего цвета и соответствует отрицательным амплитудам.

Пороговое преобразование основано на сравнении амплитуд сигналов с удвоенным пороговым уровнем шума $level$, который находится с помощью алгоритма из [1]. Если амплитуда сигнала $x_i(j)$ в канале i по модулю превысила $level$, то пикселю (i,j) в зависимости от знака амплитуды сопоставляется интенсивность красного или синего цвета по логарифмическому правилу $\left(log_{10}\frac{|x_i(j)|}{level}\right)\cdot 191 + 64$; в противном случае используется линейное правило $\frac{|x_i(j)|}{level}\cdot 64$. Каждая интенсивность округляется до целого числа. При превышении допустимого значения интенсивности полученное число понижается до 255. Гиперпараметр 64, соответствующий моменту перехода от линейной функции к

логарифмической, подобран эмпирически. В виде формулы пороговое преобразование можно описать следующим образом:

$$F(i,j) = \begin{cases} \left\lfloor \frac{|x_i(j)|}{level} \cdot 64 \right\rfloor, & |x_i(j)| < level, \\ \min\left(\left\lfloor \left(log_{10} \frac{|x_i(j)|}{level} \right) \cdot 191 + 64 \right\rfloor ; 255 \right), & |x_i(j)| \geq level. \end{cases}$$

Элементы матриц $F_r(i,j)$ и $F_b(i,j)$ формируются с помощью порогового преобразования $F(i,j)$ в соответствии со знаком амплитуды $x_i(j)$. Например, для $F_r(i,j)$ преобразование выглядит как

$$F_r(i,j) = \begin{cases} F(i,j), & \text{если } x_i(j) \geq 0, \\ 0, & \text{иначе.} \end{cases}$$

Пример изображения, полученного применением порогового преобразования к сигналу болтового стыка, приведён на рис. 4.

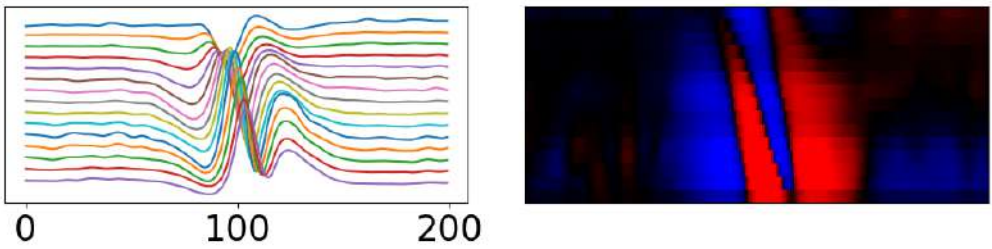


Рис. 4. Применение порогового преобразования к сигналу болтового стыка.
Fig. 4. Application of Threshold Transform to a Bolt Joint Signal.

Когда матрицы сформированы, они фрагментируются на матрицы меньшего размера с сохранением исходной высоты, но длиной 640 пикселей. Фрагменты пересекаются друг с другом. Длина пересечения равна 200. Пересечение призвано предотвратить разрез сигналов конструктивных элементов, длина которых не превышает 200 отсчётов. Для 50 тыс. отсчётов дефектограммы по каждому каналу описанная процедура позволяет получить 114 фрагментированных изображений. Из них после вертикального выстраивания с разделяющими полосами высотой 1 пиксель получаются 3 изображения размером 640×640.

3.2 Оконное преобразование Фурье

Преобразование Фурье выполняется отдельно для каждого канала дефектограммы с окном Ханна размера 64 и шагом 5. Спектр Фурье симметричен относительно нуля, то есть отрицательные и положительные частоты в спектре дублируют друг друга. В связи с этим рассматривается только положительная часть спектра. В результате по каждому каналу получается матрица размером 32×10000.

Основная часть информации о частотах полезных сигналов в вихретоковых дефектограммах представлена на отрезке угловой частоты от 0 до 1 [2]. Если рассматривать только этот диапазон частот, то размер матрицы для каждого канала сократится до 10×10000. Для того чтобы из спектра Фурье получить две матрицы (интенсивностей красного и синего цвета), красному цвету сопоставляются действительные значения гармоник, а синему – мнимые. Далее, матрицы объединяются по вертикали отдельно для каждого цвета, образуя 2 матрицы размером 150×10000. Необходимые матрицы размера 640×640 получаются с помощью описанной ранее операции фрагментации с длиной

пересечения фрагментов, равной 40. Высота разделяющих полос равна 10. Каналы из одного фрагмента соединяются друг с другом без разделения полосами. По дефектограмме длиной 50 тыс. отчётов получается 5 изображений 640×640.

3.3 Непрерывное вейвлет-преобразование

Скалограмма строится отдельно для каждого канала дефектограммы. В качестве материнской вейвлет-функции используется вейвлет Морле с параметрами $B = 2{,}5$ и $C = 0{,}5$:

$$\psi(j) = \frac{1}{\sqrt{\pi B}} e^{-\frac{j^2}{B}} e^{2i\pi C j}, i = \sqrt{-1}.$$

В качестве параметров масштаба a равномерно выбраны 8 значений из отрезка $[0; 1]$ по логарифмической шкале. Вейвлет-преобразование имеет квадратичную сложность. Для того чтобы сократить количество вычислений, амплитуды берутся с шагом 5 (сжатие в 5 раз). Если скалограммы каждого из каналов выстроить вертикально, то получится матрица размером 120×10000. Как и в случае преобразования Фурье, красный цвет формируется действительными значениями вейвлет-преобразования, а синий – мнимыми. Далее, матрицы фрагментируются с длиной пересечения фрагментов 40. Высота разделительной полосы равна 8 пикселей. Из 50 тыс. амплитуд получается 5 изображений размера 640×640.

3.4 Преобразование Гильберта-Хуанга

Для каждого канала независимо считаются внутренние модовые функции (IMF) по методу маскирующих сигналов [28], что позволяет улучшить качество эмпирической модовой декомпозиции. Далее, применяется преобразование Гильберта. Полученные спектры на каждом канале нормализуются по стандартному отклонению и суммируются друг с другом. Размер полученной матрицы сокращается в 2 раза по длине за счёт отброса каждого второго столбца матрицы (сжатие в 2 раза). Высота матрицы может быть сколь угодно большой ввиду того, что функция мгновенной частоты $\omega(t)$ является непрерывной. В нашем случае функция дискретизирована так, чтобы высота матрицы Гильбертова спектра составляла 63 пикселя.

Полученная матрица 63×25000 фрагментируется с длиной пересечения фрагментов 140. Используется только красный канал изображения, синий канал содержит нулевые значения. Зелёные разделяющие полосы имеют высоту 1 пиксель. Из дефектограммы размера 50 тыс. отсчётов получается 5 изображений.

4. Обучение YOLO

В этом разделе даётся описание того, каким образом производилось обучение модели YOLO, приводятся результаты обучения для разных видов изображений, построение которых было описано в предыдущем разделе.

4.1 Метрики оценки качества обнаружения объектов

При оценке качества обнаружения объектов необходимо контролировать как точность локализации объектов, так и надёжность их классификации. Для этого используется стандартная метрика – сбалансированная средняя точность (mean Average Precision, mAP). Она основана на понятии истинноположительных (True Positive, TP), ложноположительных (False Positive, FP) и ложноотрицательных (False Negative, FN) предсказаний модели. TP означает количество истинных предсказаний модели, FP – количество ложных предсказаний модели, а FN – количество не найденных моделью объектов. Истинность или ложность предсказания модели обычно определяется на основании вычисления пересечения над объединением (Intersection over Union, IoU), которое позволяет оценить степень

геометрической схожести отметок. Обычно для сопоставления отметок выбирается порог IoU равный 0,5.

$$\mathrm{IoU}(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

Точность (P) вычисляется как отношение количества правильных предсказаний к количеству всех предсказаний модели:

$$P = \frac{TP}{TP + FP}.$$

Полнота (R) находится как отношение количества правильных предсказаний модели к количеству действительно правильных объектов:

$$R = \frac{TP}{TP + FN}.$$

Средняя точность (AP) считается как площадь под кривой точность-полнота (PR-curve). Для формирования PR-curve предсказания модели сортируются по убыванию уверенности (confidence) предсказаний модели, затем вычисляется точность и полнота для каждого порога уверенности. Средняя точность вычисляется для каждого класса отдельно, сбалансированная средняя точность (mAP) находится как среднее арифметическое средних точностей для каждого класса. Формально это записывается следующим образом:

$$AP = \int_0^1 P(R)dR,$$

$$\mathrm{m}AP = \frac{1}{N}\sum_{i=1}^{N} AP_i,$$

где $P(R)$ – значение на RP-curve в точке R, AP_i – средняя точность для i-го класса, n – количество классов.

Число mAP50 – это та же метрика mAP, вычисленная при пороге $\mathrm{IoU} = 0{,}5$. Число mAP50-95 – это среднее арифметическое значений метрики mAP, вычисленных при пороге IoU от 0,5 до 0,95 с шагом 0,05.

4.2 Описание набора данных

Набор данных состоит из 1540 фрагментов дефектограммы длиной 50000 отсчётов каждый, на которых отмечены полезные сигналы от болтовых стыков (Bolt Joint), электроконтактных (Flash Butt Weld) и алюминотермитных (Aluminothermic Weld) сварок. Это сигналы коротких конструктивов вихретоковых данных. В зависимости от метода преобразования дефектограмм в изображения для набора данных можно получить от 4620 до 7700 изображений. В табл. 1 представлена информация о количестве изображений в обучающей, валидационной и тестовой выборках для каждого преобразования. Из набора данных исключены изображения, не содержащие отметок. Всего в наборе присутствует 4123 полезных сигнала. Информация о количестве объектов каждого класса в выборках представлена в табл. 2.

Для разных способов формирования изображений сохранялось единообразное разбиение на выборки. Вихретоковые данные изначально формируются для целых проездов – продолжительных записей одного участка железнодорожного пути. Далее, проезды нарезаются на дефектограммы по 50 тыс. отсчётов. Выборки составлялись таким образом, чтобы все участки данных, полученные из одного проезда, попадали в общую выборку. Такой подход позволяет уменьшить корреляцию между дефектограммами обучающей, валидационной и тестовой выборок и контролировать переобучение (overfitting).

Табл. 1. Размеры обучающей, валидационной и тестовой выборок.
Table 1. Sizes of Training, Validation and Test Sets.

Способ формирования изображений	**Общее количество**	**Обучающая выборка**	**Валидационная выборка**	**Тестовая выборка**
Пороговое преобразование	3360	1961	974	425
Оконное преобразование Фурье	3744	2259	1027	458
Непрерывное вейвлет-преобразование	3540	2098	1006	436
Преобразование Гильберта-Хуанга	3853	2336	1055	462

Табл. 2 Число полезных сигналов в наборе данных.
Table 2. The Number of Useful Signals in the Data Set.

Выборка	**Электроконтактные сварки**	**Алюминотермитные сварки**	**Болтовые стыки**	**Всего**
Общая	2831	1232	60	4123
Обучающая	1609	858	28	2495
Валидационная	920	184	17	1121
Тестовая	302	190	15	507

4.3 Результаты обучения

Среди сетей YOLO была выбрана одна из последних версий модели YOLOv11n. Параметры обучения перечислены в табл. 3. На рис. 5 представлены графики функции ошибки во время обучения на обучающей и валидационной выборках.

Табл. 3. Гиперпараметры обучения YOLO.
Table 3. Hyperparameters of YOLO Training.

Параметр	**Значение**
Количество эпох	300
Размер пакета	30
Оптимизатор	AdamW
Начальный темп обучения	0,00149
Инерция	0,9

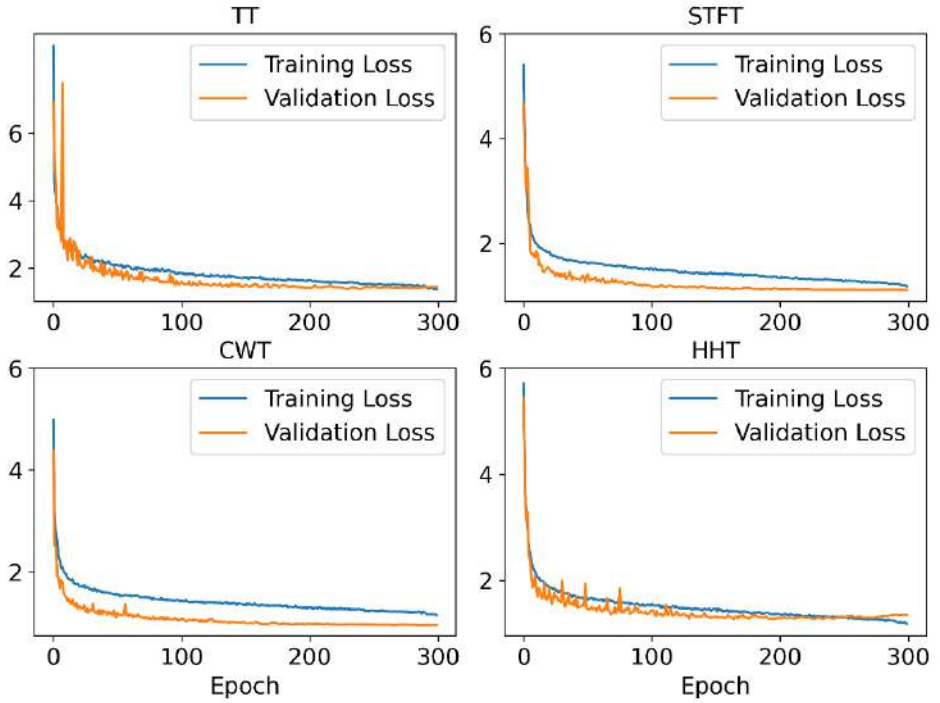


Рис. 5. Графики функции ошибки на обучающей и валидационной выборках.
Fig. 5. Training and Validation Loss Curves.

В табл. 4 отражены значения сбалансированной средней точности mAP50 и mAP50-95 для обученных моделей YOLO на разных видах изображений. В целом, все виды преобразований сигналов в изображения в интеграции с YOLO позволяют получить высокие показатели точности обнаружения полезных сигналов вихретоковых дефектограмм.

Табл. 4 Сбалансированная средняя точность mAP обученных моделей YOLO.
Table 4. Mean Average Precision (mAP) of Trained YOLO Models.

Способ формирования изображений	**mAP50**	**mAP50-95**
Пороговое преобразование	0,9792	0,9694
Оконное преобразование Фурье	0,9790	0,9656
Непрерывное вейвлет-преобразование	0,9823	0,9742
Преобразование Гильберта-Хуанга	0,9292	0,9203

Лучшие результаты показывает вейвлет-преобразование. Пороговое преобразование выигрывает у других методов простотой вычислений и линейной трудоёмкостью. На создание одного изображения ушло 372 мс для STFT, 97 мс – CWT, 9716 мс – HHT и всего 61 мс – для порогового преобразования TT. Среди частотно-временных методов STFT имеет близкие к CWT показатели mAP, тогда как HHT им немного уступает. Это можно объяснить используемым подходом к формированию гильбертова спектра многоканального сигнала,

когда каналы объединяются простым усреднением спектра и не используется синий канал RGB формата изображения. Видимо, следует рассмотреть более надёжные методы HHT, например, HHT с Multidimensional Ensemble Empirical Mode Decomposition (MEEMD) [29]. Также за счёт RGB формата можно соединить несколько частотных представлений сигнала в одном изображении, например, как в работе [16].

Матрицы ошибок (рис. 6) показывают, что при использовании порогового преобразования сеть, несмотря на высокие показатели mAP, делает наибольшее число ложноположительных (FP) предсказаний по сравнению с остальными преобразованиями. Наименьшее число ложноположительных предсказаний происходит при интеграции сети YOLO с STFT. При этом преобразование Фурье практически не уступает другим методам по числу истинноположительных предсказаний (TP), а в случае электроконтактных сварок показывает наилучшие результаты. Преобразование Гильберта-Хуанга хуже остальных находит полезные сигналы и многие из них пропускает. Пороговое преобразование лучше реагирует на алюминотермитные сварки. Таким образом, с точки зрения матрицы ошибок использование STFT по сравнению с другими преобразованиями обладает рядом преимуществ: происходит наименьшее число ложных предсказаний, находятся оказавшиеся наиболее сложными для обнаружения сетью электроконтактные сварки и практически не пропускаются другие полезные сигналы.

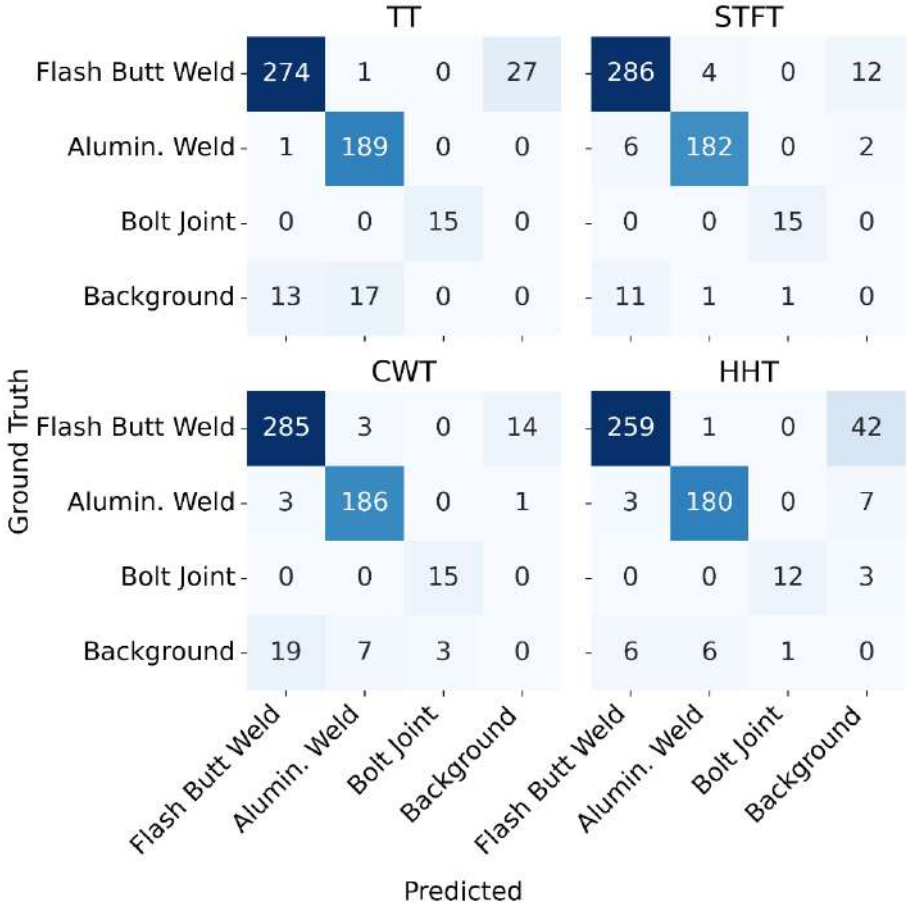


Рис. 6. Матрицы ошибок обученных моделей YOLO.
Fig. 6. The Confusion Matrixes of Trained YOLO Models.

На рис. 7 представлены примеры работы сетей YOLO, интегрированных с каждым из рассмотренных преобразований, на фрагменте дефектограммы с рис. 1.

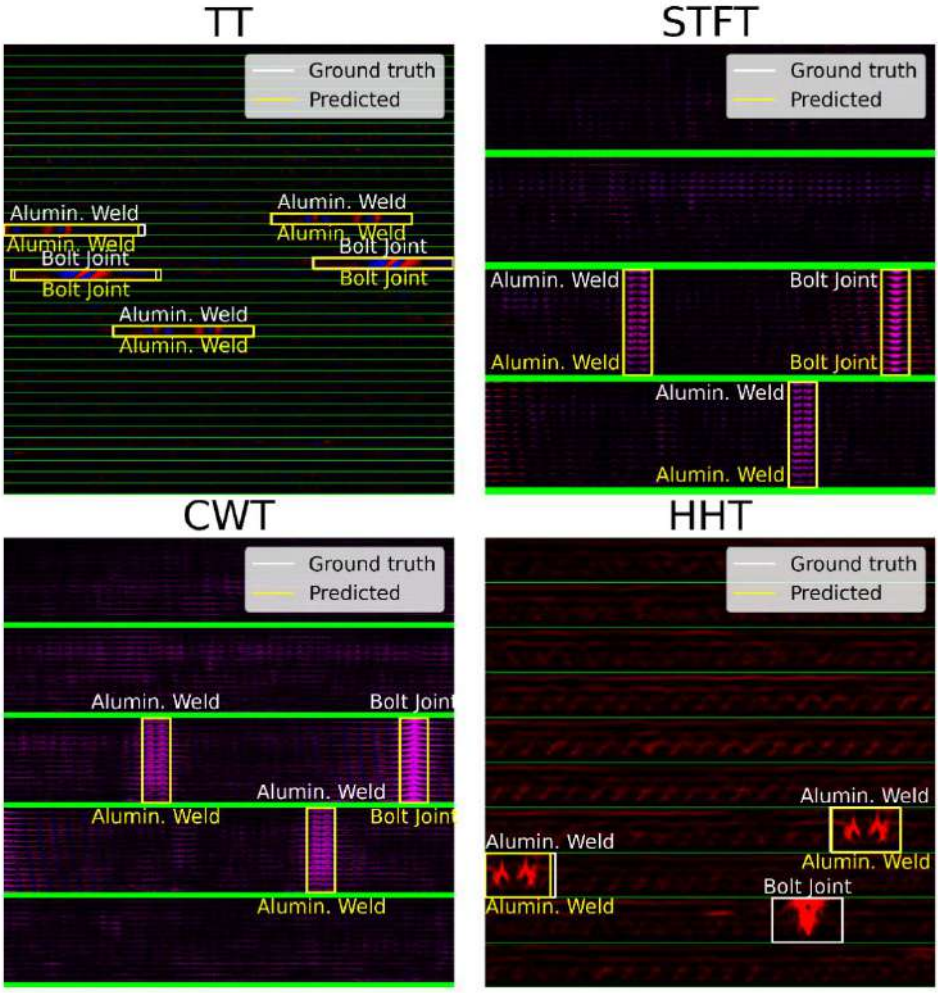


Рис. 7. Примеры работы обученных моделей YOLO.
Fig. 7. The Example of Inference of Trained YOLO Models.

5. Заключение

Задача обнаружения полезных сигналов может решаться разными методами. Один из подходов основан на свёрточных нейронных сетях, которые в последнее время стали основным инструментом в решении задач компьютерного зрения. Для применения этого подхода сигналы необходимо преобразовать в изображения. Существует множество способов выполнить такое преобразование. В данной статье применительно к вихретоковым дефектограммам рельсов, которые представляют собой 15-канальный дискретный сигнал, рассмотрены оконное преобразование Фурье (STFT), непрерывное вейвлет-преобразование (CWT), преобразование Гильберта-Хуанга (HHT) и пороговое преобразование (TT). Все методы кроме TT строят частотно-временной спектр сигналов. Предложенное авторами пороговое преобразование основано на простом сопоставлении каждой амплитуде сигнала значения интенсивности цвета пикселя.

Рассмотренные преобразования можно интегрировать вместе с детектирующей свёрточной нейронной сетью YOLO. В результате было обучено 4 сети YOLO для изображений каждого вида. Наименьшими временными затратами на формирование изображений обладает пороговое преобразование. Наибольшие значения mAP показала сеть на основе CWT. Лучший баланс между точностью и полнотой нахождения полезных сигналов показала сеть на основе STFT. В целом, каждое из преобразований позволило получить высокие показатели метрик.

Таким образом, для вихретоковых дефектограмм могут использоваться разные способы формирования изображений в зависимости от требований на время выполнения, количество пропусков и ложных срабатываний, значимости полезных сигналов определённого класса и т. д. Результаты исследования показывают перспективность применения сетей семейства YOLO при решении задачи обнаружения полезных сигналов вихретоковых дефектограмм и одномерных данных в целом. В дальнейшем планируется рассмотреть подходы к обнаружению полезных сигналов на основе других видов изображений и с использованием одномерной детектирующей свёрточной сети, построенной на основе архитектуры YOLO, для которой не будет требоваться трансляция сигналов в изображения.

Список литературы / References

[1]. Kuzmin E. V., Gorbunov O. E., Plotnikov P. O., Tyukin V. A. Finding the Level of Useful Signals on Interpretation of Magnetic and Eddy-Current Defectograms. Automatic Control and Computer Sciences, vol. 52, 2018, pp. 658-666. DOI: 10.3103/S0146411618070179.

[2]. Быстров Л. Ю., Гладков А. Н., Кузьмин Е. В. Подавление аддитивных периодических низкочастотных помех на вихретоковых дефектограммах. Моделирование и анализ информационных систем, 31(2), с. 164-181. DOI: 10.18255/1818-1015-2024-2-164-181. / Bystrov L.Y., Gladkov A.N., Kuzmin E.V. Suppression of additive periodic low-frequency interference on eddy current defectograms. Modeling and Analysis of Information Systems, 31(2), 2024, pp. 164-181 (In Russian).

[3]. Chao-Lung Y., Zhi-Xuan C., Chen-Yi Y. Sensor Classification Using Convolutional Neural Network by Encoding Multivariate Time Series as Two-Dimensional Colored Images. Sensors, 20(1), 2019, article no. 168. DOI: 10.3390/s20010168.

[4]. Terven J., Cordova-Esparza, D. M., Romero-González J. A. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Machine Learning and Knowledge Extraction, 5(4), 2023, pp. 1680-1716. DOI: 10.3390/make5040083.

[5]. Hussain M. YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. Machines, 11(7), 2023, article no. 677. DOI: 10.3390/machines11070677.

[6]. Wyse L. Audio Spectrogram Representations for Processing with Convolutional Neural Networks. 2017. DOI: 10.48550/arXiv.1706.09559. – Available at: https://arxiv.org/abs/1706.09559.

[7]. Junior R. F. R. et al. Fault Detection and Diagnosis in Electric Motors Using Convolution Neural Network and Short-Time Fourier Transform. Journal of Vibration Engineering & Technologies, vol. 10, 2022, pp. 1-12. DOI: 10.1007/s42417-022-00501-3.

[8]. Modupalli V. P. CNN-Based Machine Tool Monitoring with STFT Image Analysis. International Journal for Research in Applied Science and Engineering Technology, 12(5), 2024, pp. 3986-3991. DOI: 10.22214/ijraset.2024.62493.

[9]. Yu Y., Liu Q., Han B. S., Zhou W. Application of Convolutional Neural Network to Defect Diagnosis of Drill Bits. Applied Sciences, 12(21), 2022, article no. 10799, 13 p. DOI: 10.3390/app122110799.

[10]. Cheng C., Seo H., Zhao Y. A Novel Pavement Transverse Cracks Detection Model Using WT-CNN and STFT-CNN for Smartphone Data Analysis. International Journal of Pavement Engineering, 23(12), 2021, pp. 1-13. DOI: 10.1080/10298436.2021.1945056.

[11]. Kang X. et al. Optimal Preprocessing for Joint Detection and Classification of Wireless Communication Signals in Congested Spectrum Using Computer Vision Methods. 2024. DOI: 10.48550/arXiv.2408.06545. – Available at: https://arxiv.org/abs/2408.06545.

[12]. Liu C., Wang D., Lin Y., Song, S. Research on Online Monitoring of Chatter Based on Continuous Wavelet Transform and Convolutional Neural Network – Vision Transformer (CNN-ViT). Mechanical Sciences, 16(1), 2025, pp. 167-180. DOI: 10.5194/ms-16-167-2025.
[13]. Lupea I., Lupea M. Continuous Wavelet Transform and CNN for Fault Detection in a Helical Gearbox. Applied Sciences, 15(2), 2025, article no. 950, 23 p. DOI: 10.3390/app15020950.
[14]. Pałczyński C., Olejnik P. Anomalies Classification in Fan Systems Using Dual-Branch Neural Networks with Continuous Wavelet Transform Layers: An Experimental Study. Information, 16(2), 2025, article no. 71, 13 p. DOI: 10.3390/info16020071.
[15]. Piedad E. J., Rosario C., Prieto-Araujo E., Bellmunt O. Exploring Wavelet Transformations for Deep Learning-Based Machine Condition Diagnosis. In Proc. of the International Conference on Diagnostics in Electrical Engineering (Diagnostika), 2024, pp. 1-4. DOI: 10.1109/Diagnostika61830.2024.10693895.
[16]. Siddique M. F, Ahmad Z, Ullah N, Kim J. A Hybrid Deep Learning Approach: Integrating Short-Time Fourier Transform and Continuous Wavelet Transform for Improved Pipeline Leak Detection. Sensors (Basel), 23(19), 2023, article no. 8079. DOI: 10.3390/s23198079.
[17]. Ch Vidyasagar K. E. et al. Signal to Image Conversion and Convolutional Neural Networks for Physiological Signal Processing: A Review. IEEE Access, vol. 12, 2024, pp. 66726-66764. DOI: 10.1109/ACCESS.2024.3399114.
[18]. Alvarenga T. A. et al. Detection and Classification System for Rail Surface Defects Based on Eddy Current. Sensors, 21(23), 2021, article no. 7937. DOI: 10.3390/s21237937.
[19]. Sagar B., Kane P. Intelligent Bearing Fault Diagnosis of Switched Reluctance Motor Using Hilbert-Huang Transformation and Region-Based Convolutional Neural Network. The International Journal of Acoustics and Vibration, 29(4), 2024, pp. 431-441. DOI: 10.20855/ijav.2024.29.42072.
[20]. Srivastava S. et al. Comparative analysis of deep learning image detection algorithms. Journal of Big Data, 8(66), 2021. DOI: 10.1186/s40537-021-00434-w. Available at: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00434-w.
[21]. Po-Liang Y., Pei-Ling L. Automatic Detection of Crack Echo Signals in Spectrogram Using Yolo. Available at: https://ssrn.com/abstract=4885492. DOI: 10.2139/ssrn.4885492.
[22]. Vargas V. M. et al. Gramian Angular and Markov Transition Fields Applied to Time Series Ordinal Classification. Advances in Computational Intelligence, 2023, pp. 505-516. DOI: 10.1007/978-3-031-43078-7_41.
[23]. Zhang Y., Chen X. Motif Difference Field: A Simple and Effective Image Representation of Time Series for Classification. 2020. DOI: 10.48550/arXiv.2001.07582. – Available at: https://arxiv.org/abs/2001.07582.
[24]. Jun-Hao C., Yun-Cheng T. Dynamic Deep Convolutional Candlestick Learner. 2022. DOI: 10.48550/arXiv.2201.08669. – Available at: https://arxiv.org/abs/2201.08669.
[25]. Dan-Feng W. et al. Planetary-Gearbox Fault Classification by Convolutional Neural Network and Recurrence Plot. Applied Sciences, 10(3), 2020, article no. 932. DOI: 10.3390/app10030932.
[26]. Som A. et al. PI-Net: A Deep Learning Approach to Extract Topological Persistence Images. In Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 3639-3648. DOI: 10.1109/CVPRW50498.2020.00425.
[27]. Samani E. U., Banerjee A. G. Persistent Homology Meets Object Unity: Object Recognition in Clutter. In Proc. of the IEEE Transactions on Robotics, vol. 40, 2024, pp. 886-902. DOI: 10.1109/TRO.2023.3343994.
[28]. Yang Y., Deng J., Kang D. An Improved Empirical Mode Decomposition by Using Dyadic Masking Signals. Signal, Image and Video Processing, vol. 9, 2015, pp. 1259-1263. DOI: 10.1007/s11760-013-0566-7.
[29]. Zhaohua W., Jiaxin F., Fang-Li Q., Zhe-Min T. Fast Multidimensional Ensemble Empirical Mode Decomposition for the Analysis of Big Spatio-Temporal Datasets. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), 2016, article no. 20150197. DOI: 10.1098/rsta.2015.0197.

Информация об авторах / Information about authors

Артемий Николаевич ГЛАДКОВ – ассистент кафедры теоретической информатики. Научные интересы: цифровая обработка сигналов, машинное обучение, прикладная статистика.

Artemy Nikolaevich GLADKOV – assistant of the Department of Theoretical Computer Sciences. Research interests: digital signal processing, machine learning, applied statistics.

Леонид Юрьевич БЫСТРОВ – ассистент кафедры теоретической информатики. Научные интересы: цифровая обработка сигналов, машинное обучение, топологические методы анализа динамических систем, математическая статистика.

Leonid Yurievich BYSTROV – assistant of the Department of Theoretical Computer Sciences. Research interests: digital signal processing, machine learning, topological methods of dynamic systems analysis, mathematical statistics.

Егор Владимирович КУЗЬМИН – доктор физико-математических наук, доцент, заведующий кафедрой теоретической информатики. Научные интересы: цифровая обработка сигналов, методы верификации программ и теоретические основы информатики.

Egor Vladimirovich KUZMIN – Dr. Sci. (Phys.-Math.), Associate Professor, Head of the Department of Theoretical Computer Sciences. Research interests: digital signal processing, program verification methods and theoretical foundations of computer sciences.

DOI: 10.15514/ISPRAS-2025-37(6)-26

Инструменты платформы ЛингвоДок в изучении гидронимов Республики Саха (Якутия)

М.В. Самсонова, ORCID: 0000-0001-5637-4085 <mv.samsonova@s-vfu.ru>

Северо-Восточный федеральный университет имени М.К. Аммосова,
677000, Россия, Республика Саха (Якутия), г. Якутск, ул. Белинского, д. 58.

Аннотация. В статье представлены принципы составления словаря имен собственных водных объектов (гидронимов) Республики Саха (Якутия) для дальнейшей работы с ним на платформе ЛингвоДок. В работе над словарем применяется комплексный подход с точки зрения лексикографии, лексикологии, семантики, морфологии, этимологии, картографии. Описана методология отбора и анализа топонимического материала, описаны проблемы искажения названий при картографировании по правилам русского языка, основные структурные типы гидронимов, принципы выделения семантических признаков, разделения их на группы. Представлены карты, созданные на основе данных словарей топонимов, загруженных на платформу ЛингвоДок. Словарь гидронимов является первой попыткой систематизации названий водных объектов Республики Саха (Якутия) на платформе ЛингвоДок.

Ключевые слова: топонимия; гидронимы; Республика Саха (Якутия); якутский язык (язык саха); платформа ЛингвоДок; топонимическая номинация; мотивационный признак.

Для цитирования: Самсонова М.В. Инструменты платформы ЛингвоДок в изучении гидронимов Республики Саха (Якутия). Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 151–168. DOI: 10.15514/ISPRAS–2025–37(6)–26.

LingvoDoc Platform Tools in the Study of Hydronyms of the Republic of Sakha (Yakutia)

M.V. Samsonova, ORCID: 0000-0001-5637-4085 <mv.samsonova@s-vfu.ru>

North-Eastern Federal University named after M.K. Ammosova,
58, Belinskogo street, city of Yakutsk, Republic of Sakha (Yakutia), 677000, Russia.

Abstract. The article presents the principles of compiling a dictionary of hydronyms, proper names of water bodies in the Republic of Sakha (Yakutia), for further work with it on the LingvoDoc platform. The dictionary is compiled using a comprehensive approach from the perspectives of lexicography, lexicology, semantics, morphology, etymology, and cartography. The article describes the methodology of selecting and analyzing toponymic material, the problems of distortion of names during mapping according to the rules of the Russian language, the main structural types of hydronyms, and the principles of identifying semantic features and dividing them into groups. The article presents maps created based on the data of toponym dictionaries uploaded to the LingvoDoc platform. The hydronym dictionary is the first attempt to systematize the names of water bodies in the Republic of Sakha (Yakutia) on the LingvoDoc platform.

Keywords: toponymy; hydronyms; Republic of Sakha (Yakutia); LingvoDoc platform; Yakut (Sakha) language; toponymic nomination; motivational feature.

For citation: Samsonova M.V. LingvoDoc platform tools in the study of hydronyms of the Republic of Sakha (Yakutia). Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 151-168 (in Russian). DOI: 10.15514/ISPRAS-2025-37(6)-26.

1. Введение

Разработка лингвистической платформы ЛингвоДок [1] Институтом системного программирования РАН в тесном сотрудничестве с ведущими учеными Института языкознания РАН стала своевременным ответом на растущие запросы российских исследователей по цифровой обработке, структурированию, документированию и картографированию языкового материала на языках народов России. Основные возможности, предоставляемые платформой ЛингвоДок пользователям, это работа со словарями, работа с корпусами текстов и картографирование лингвистических особенностей. Платформа позволяет быстро провести анализ объемного материала, картографировать, построить ареалы распространения разных языковых явлений, дает возможность наглядно проиллюстрировать результаты исследования, благодаря опции поиска данных на карте с автоматическим построением изоглосс с возможностью скачивания результатов поиска в формате Excell. Многопользовательская платформа дает возможность проводить в том числе сравнительно-сопоставительные исследования, привлекая материал, загруженный другими пользователями.

Решение исследовать из всего многообразия топонимов Республики Саха (Якутия) именно гидронимы, связано в первую очередь с тем, что в настоящее время назрела необходимость в комплексном труде по топонимике Республики Саха (Якутия) в цифровом формате. Еще одним важным фактором является то, что для исследования топонимов очень важна возможность работы с картами. Карты, которые можно построить на платформе ЛингвоДок, работают с координатами сервиса Яндекс Карты – поисково-информационной картографической службой Яндекса [2]. Данный сервис является основным картографическим сервисом для российских пользователей в настоящее время, он очень удобен для использования в населенных пунктах, все данные постоянно обновляются, карты доступны в трех вариантах (схема, спутник, гибрид), есть возможность просмотра улиц и т.д. Несмотря на то, что карты сервиса не очень подробно отражают названия природных

географических объектов, которые находятся за пределами населенных пунктов, в сервисе из всех природных объектов лучше всего отображены названия водных объектов - гидронимы.

По данным Якутского управления по гидрометеорологии и мониторингу окружающей среды на территории республики расположено около 700 тысяч рек и речек, из которых более 29 тысяч рек длиной более 10 км, и более 637 тысяч озер [3]. В Яндекс Карты отображены названия большинства крупных и средних озер на территории Российской Федерации.

Под гидронимом в работе понимается вслед за Н.В. Подольской собственное имя любого водного объекта, природного или созданного человеком, в том числе океаноним (название океана), пелагоним (название моря), лимноним (название озера), потамоним (название реки), гелоним (название болота) [4]. Гидронимы относятся к наиболее древним топонимам, о чем свидетельствуют труды исследователей [5], [6]. Водные объекты имеют огромную важность для жизни человека, являясь главными ориентирами в пространстве при освоении территории, местом расселения народов, так как реки и озера являются источником пресной воды, местом рыбалки и охоты и транспортными артериями. Часто значение имени собственного реки или озера уже утрачено в наше время, так как топонимическая стратиграфия почти любой территории обнаруживает временные наслоения, и многие из древних гидронимов были заимствованы у предшествующих народов.

В настоящей статье внимание будет уделено гидронимам с зоокомпонентом, т.е. названием представителя животного мира Республики Саха (Якутия), отличающегося большим разнообразием.

Материал исследования, включающий 1000 гидронимов, был отобран из разных справочников, карт, исследований топонимики Республики Саха (Якутия), в частности, трудов основоположника якутской топонимики М.С. Иванова – Багдарыына Сулбэ, собравшего и проанализировавшего огромный полевой материал по топонимам на языках народов республики: якутском, эвенкийском, эвенском, юкагирском, чукотском и русском с привлечением данных фольклора, литературы, свидетельств информантов, архивных данных, картографии [7-9]. Список может быть пополнен трудами Н.М. Иванова [10], Ю.А. Слепцова [11], Ю.Г. Курилова [12] и др. Обязательным условием при отборе материала для исследования было наличие данного гидронима в сервисе Яндекс Карты, так как далее географические координаты водных объектов из данного сервиса используются при построении подробных карт на платформе ЛингвоДок.

2. Разработка и подготовка словаря для загрузки на платформу ЛингвоДок

Для создания словаря в ЛингвоДок необходимо было ввести на платформу данные в табличной форме в формате CSV. Для анализа отобранного материала из 1000 гидронимов на якутском языке была создана таблица и разбита на ряд колонок, которые подробно описывают каждый гидроним по следующим критериям: тип гидронима, написание гидронима на якутском языке, дословно переведенное название на русский язык с опорой на словари и другие источники, картографированное название на русском языке, способ образования топонима, этимология, мотивационный признак, географические координаты. Ниже мы рассмотрим каждый критерий подробно. Данный список может быть дополнен другими критериями в зависимости от целей исследования.

2.1 Тип топонима, написание топонима на якутском языке и буквальный перевод на русский язык

При выделении типа гидронима, мы опирались на классификацию топонимов, разработанную в трудах Н.В. Подольской [4], Э.М. Мурзаева [6], А.В. Суперанской [13], Р.А. Агеевой [3] и др. В работе исследуются следующие разновидности гидронимов: лимнонимы

– названия озер и потамонимы - названия рек. В Яндекс Карты можно найти названия почти всех рек и речек и многих крупных и средних озер Республики Саха (Якутия). В материал исследования включены также инсулонимы, названия островов, которые нанесены в Яндекс Карты. Возможность построения топонимических карт с изоглоссами в ЛингвоДок на основе введения точных координат водных объектов из Яндекс Карты является одной из опций, позволяющих наглядно проиллюстрировать распространение различных топонимических явлений. Под топонимическими картами в работе понимается тематическая карта, показывающая распространение элемента топонима, отдельного топонима, группы топонимов, топонимического явления, динамики его развития, его связи с социальными явлениями [4].

Написание топонима очень важно для дальнейшего анализа топонима, так как при картографировании часто идет сильное искажение оригинального названия так, что порой с трудом можно понять, как картографированный топоним звучит на языке автохтонов территории и что он означает. В 1976 году главным управлением геодезии и картографии при Совете Министров СССР была издана инструкция по русской передаче географических названий Якутской АССР Г.И. Донидзе, которая служила основой для ассимиляции названия на якутском языке по правилам русского языка [14]. Инструкция снабжена краткими сведениями о якутском языке и якутских географических названиях, отличается стремлением к минимальному искажению оригинального звучания топонима. Но не все топонимы Якутии, ассимилированы согласно этой инструкции, так, например, название реки Иткин-Сиебиттир в Алданском районе из-за сильного искажения непонятна носителям якутского языка. На якутском языке данный гидроним звучит как *Ыккын Сиэбиттэр* букв. ‘съели/убили твою собаку’. В вышеозначенной инструкции по передаче географических названий с якутского на русский, замена *ы* в якутском слове на *и* при ассимиляции в русском возможна только после *дь* и *нь*, которых здесь нет. Гидроним должен был быть ассимилирован как «Ыккын-Сиебиттэр». Еще одной проблемой при передаче якутских топонимов может быть ассимиляция гласных звуков, не существующих в русском языке, например, звука *ө* [ø], который передается в русском через *ё*. Из-за того, что две точки над *ё* часто не печатаются на картах, в ассимилированных названиях непонятно, какой звук там стоит в оригинальном звучании. Возникает ложная омонимия, как например, картографированное название “оз. Бере” может означать в якутском *Бэрэ* ‘небольшое оз., прилегающее к другому, большему озеру’ или *Бөрө* ‘волк’, в котором лучше ставить точки над ё: озеро Бёрё.

Буквальный перевод на русский язык позволяет широкому кругу пользователей платформы понять значение топонима, увидеть механизм создания онима, проследить логику, которой руководствовался человек, давший название данному географическому объекту. Основными лексикографическими источниками при толковании значения являются толковый словарь якутского языка П.А.Слепцова [15], словарь якутского языка Э.К.Пекарского [16], а также современный онлайн сервис https://sakhatyla.ru/.

Столбец «Картографическое название» содержит название гидронима, занесенное в Яндекс Карты в ассимилированном фонетикой русского языка виде с указанием района (улуса). Например, «оз. Кыл-Муостах, Олёкминский муниципальный район, Республика Саха (Якутия)», на якутском *Кыыл Муостаах* ‘где имеются рога дикого оленя/лося’. Указание улуса в словаре позволит сделать выводы о распространенности, частотности разных топонимических явлений, особенностях топонимической номинации в 34 улусах республики, имеющих каждый свою историю, этнические и языковые контакты, различную топонимическую стратиграфию. Топонимическая стратиграфия – последовательность временных напластований и пространственного взаимоотношения комплексов географических названий на данной территории [4].

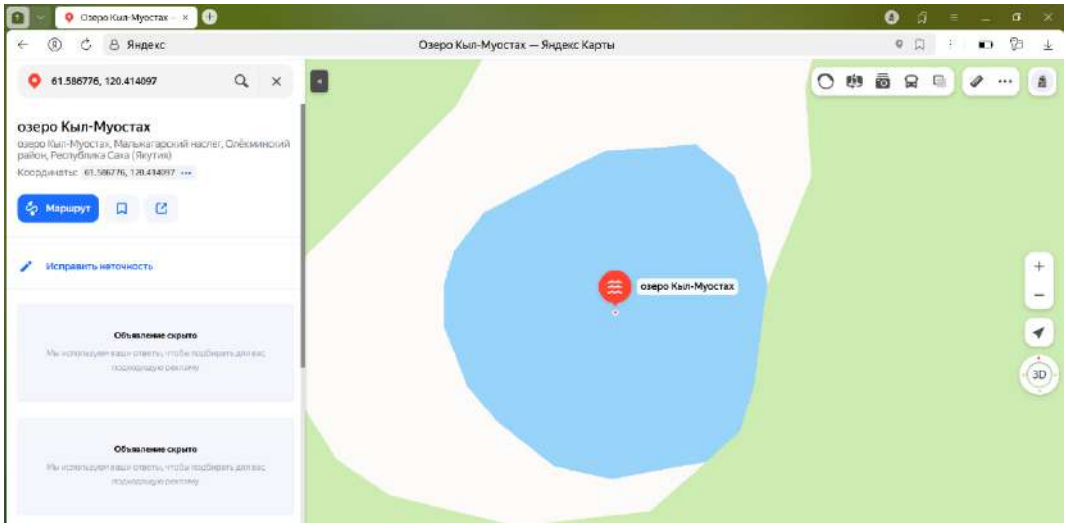


Рис. 1. Пример картографированного ассимилированного русским языком якутского топонима в сервисе Яндекс Карты с указанием района, субъекта РФ, точными географическими координатами.
Fig. 1. An example of a map of the Yakut toponym assimilated in Russian in the Yandex Maps service with an indication of the region, the subject of the Russian Federation, and exact geographical coordinates.

2.2 Морфологические и структурные особенности гидронимов

В разделе указывается способ образования гидронима. Наиболее часто в отобранном материале встречается основной способ аффиксации в якутском языке, заключающийся в присоединении к основам специальных словообразующих и словоизменяющих аффиксов. Словообразующие аффиксы дополняют или изменяют лексическое значение основы и образуют от нее производную основу, выступающую как новое самостоятельное слово, которое может принимать различные аффиксы: якут. *балык ‘рыба’* афф. – лаа (-таа) образует производный глагол *балыктаа- ‘ловить рыбу, рыбачить’,* который может дальше принимать различные аффиксы. В топонимах чаще встречается аффиксация при помощи словоизменяющих аффиксов, которые не меняют лексическое значение основы, а служат для выражения связи или отношения данного слова к другим словам, входящим в состав словосочетания. Такие аффиксы образуют окончательную форму слова, к которой уже нельзя применить другие аффиксы [17].

В настоящем исследовании в таблице гидронимов обозначено два способа образования топонимов, морфологический и синтаксический. Морфологический способ является очень продуктивным в топонимах, например, часто встречается аффикс обладания *-лаах* и его варианты: оз. Балыктах/ якут. *Балыктаах* (‘с рыбой/ где водится рыба’); река Бёрёлёх/ якут. *Бөрөлөөх* (‘с волком/ где водятся волки’); оз. Куобахтах/ якут. *Куобахтаах* (‘с зайцем/ где есть зайцы’), оз. Кырынастах/ якут. *Кырынаастаах* (‘с горностаем/ где есть горностай’), оз. Эсэлях/ якут. *Эһэлээх* (‘медведем/ где есть медведь’), оз. Табалах/ якут. *Табалаах* (‘с оленем/ где есть олень’), оз. Майагастах/ якут. *Майаҕастаах* (‘с пыжьяном/ где есть пыжьян’) и др. Всего в отобранном материале насчитывается 409 гидронимов данного типа, что составляет 41% от общего количества. Особый подвид гидронимов, состоящий из глагола, в основе которого лежит название животного с аффиксом действия *-ыыр* (*-иир*, *-уур*, *-үүр*): оз. Соболур/ *Соболуур* (‘где ловят карася’), оз. Сордоннур/ *Сордоҥнуур* (‘где ловят щуку’), оз. Кустур/ *Кустуур* (‘где охотятся на утку’), оз. Андылыр/ *Андылыыр* (‘где охотятся на турпана’) содержит в своей семантике указание на то, что в данном водном объекте водится в изобилии тот или иной вид.

Вторым способом образования является синтаксический, посредством сочетания слов: оз. Ыт-Кюеле/ якут. *Ыт Күөлэ* (*ыт* ‘собака’ и *күөл* ‘оз.’+аффикс принадлежности -э *букв. ‘оз. собаки’),* река Кырса-Ыйабыт/ якут. *Кырса Ыйаабыт* (кырса ‘*песец’* и ыйаа ‘повесить’+ окончание причастия прошедшего времени -*быт,* таким образом получаем буквальное значение *‘где повесили песца’*). Компонент *ыйаабыт,* при картографировании теряет одну *а* или пишется через *я* «ыябыт”, он является одним из самых частотных в якутской топонимии: протока Ыт-Ыйабыт/ *Ыт Ыйаабыт* ‘где повесили собаку’, оз. Кырса-Ыйабыт/ *Кырса Ыйаабыт* ‘где повесили песца’, оз. Ангыр-Ыябыт/ якут. *Анҕыр Ыйаабыт* ‘где повесили выпь’ и т.д. У Багдарына Сулбэ находим, что в целях сохранения, часто подвешивали таким образом как вещи, так и припасы на деревья, чтобы не достал зверь, не замело снегом и т.д. Также, таким способом обозначали ничем не примечательные места, что-то вешалось специально в качестве указателя, ориентира [9]. Весьма распространены в Якутии топонимы с глаголом прошедшего времени со сходным аффиксом -*быт*, часто они отмечают какое-то памятное, трагическое событие, связанное со смертью домашнего или дикого животного: оз. Тугут-Сюппют/ *Тугут Сүппүт* (‘пропал олененок’), оз. Кыл-Тимирбит/ *Кыыл Тимирбит* (‘дикий олень/лось утонул’), река Улахан-Ыт-Ольбют/ *Улахан Ыт Өлбут* (‘большая (река) собака умерла’) и др. Потеря верного пса для охотника или гибель домашнего скота для скотовода были, конечно, весьма значимыми событиями, тем более, если это произошло в условиях экстремально низких температур зимы в Якутии, что послужило поводом для появления данного названия. Понимание этого факта, отраженное в топонимике, передает значимость охоты и скотоводства для выживания, как основных видов хозяйствования на данной территории.

Среди отобранных гидронимов с зоокомпонентом нам встретились однокомпонентные (40%), двухкомпонентные (52%), трехкомпонентные гидронимы (7%) и четырехкомпонентные (1%).

Самый многочисленный вид гидронимов - двухкомпонентные гидронимы, отличаются большим разнообразием, вторым компонентом, помимо зоокомпонента, в них могут быть как существительные, обозначающие водный объект (озеро, река, речка, остров), типа, оз. Кистях-Кель/ *Киистээх-Күөл* ‘оз. с соболем’, река Ат-Юрюе/ *Ат Үрүйэ* ‘речка конь’, или какой-то связанный с животным факт, типа оз. Бёрё-Уялах/ *Бөрө Уйалаах* ‘с норой волка’, река Бёре-Улуйбут| *Бөрө Улуйбут* ‘волк выл’, и даже лекарственное растение: оз. Бёрё-Оттох/ *Бөрө Оттоох* ‘с волчьей травой’. Вторым компонентом могут быть прилагательные, чаще уточняющие положение озера или его размер относительно озера с таким же названием, по существующей в топонимике бинарной оппозиции: оз. Уеся-Кыллах/ *Уөһээ Кыыллаах* (‘верхнее (озеро) со зверем/ с диким оленем’) и оз. Аллара-Кыллах/ *Аллараа Кыыллаах* (‘нижнее (озеро) со зверем/ с диким оленем’), река Арга-Быйыттах (‘западная (река) с острорылым ленком’) и Илин-Быйыттах/ *Илин Быйыттаах* (‘восточная (река) с острорылым ленком’), оз. Аччыгый-Кистях/ *Аччыгый Киистээх* (‘малое с соболем’) и Улахан-Кистях/ *Улахан Киистээх* (‘большое с соболем’). Встречаются также гидронимы с числительными оз. Алта-Соболох/ *Алта Соболоох* (‘шесть карасей’) и цветом рыбы, типа оз. Сасыл-Мундулах/ *Саһыл Мундулаах* (‘с желтым гольяном озерным’). Часто в якутских гидронимах с зоокомпонентом вторым компонентом выступает лексема с отрицанием *суох* ‘нет; без’, типа оз. Балыга Суох (‘без рыбы’) или оз. Атага-Суох/ *Атаҕа Суох* (‘озеро без залива’).

В трехкмопонентных гидронимах нам встретились все типы, которые встречаются в двух-компонентных, в них третьим элементом часто выступает географический термин, обозначающий тип водоема, типа оз. Бёрё-Уялах-Кюель /*Бөрө Уйалаах Күөл* ‘оз. с норой волка’, река Огус-Суоллах-Юрях/ *Оҕус Суоллаах Үрэх* ‘речка со следом быка’, или это двухкомпонентные гидронимы с прилагательными, уточняющими положение озера или его размер относительно другого озера с таким же названием, типа оз.Аччыгый-Хара-Соболох/ *Аччыгый-Хара-Соболоох* ‘малое с черным карасем’, оз. Илин-Ат-Асатар/ *Илин Ат Аһатар* ‘восточное, где кормят лошадей’.

Четырехкомпонентные гидронимы часто содержат прилагательное размера, местоположения и двухкомпонентное название местности на которой расположен тот или иной водный объект, и также содержат четвертый компонент, выраженный географическим термином, обозначающим сам водный объект, особенно это касается инсулонимов, названий островов: остров Оччугуй-Кыл-Муостах-Бёлькёё/ *Оччугуй Кыыл Муостаах Бөлкөйө* ‘малый остров (протоки) с рогами зверя’, рядом расположен Улахан-Кыл-Муостах-Бёлькёё/ *Улахан Кыыл Муостаах Бөлкөйө* ‘большой остров (протоки) с рогами зверя’; остров Кыллах-Хара-Сырын-Бёлькёё/ *Кыыллаах-Хара-Сыырын-Бөлкөйө* ‘остров черной горы местности с диким оленем/ лосем’, чаще это наблюдается у островов, не имеющих собственного названия.

2.3 Происхождение гидронимов и их мотивационные признаки

В данном разделе указывается этимология зоокомпонента гидронима. Все зоокомпоненты в отобранном гидронимиконе на якутском языке можно поделить на 4 основные группы: “Рыбы” (282 ед.), “Птицы” (272 ед.), “Домашние животные” (247 ед.), “Дикие животные” (205 ед.). Предварительная работа с этимологическими основами зоокомпонентов отобранных гидронимов показывает, что большинство названий животных, птиц и рыб в якутском языке, имеет тюркское происхождение: *бөрө* волк от PTurk. *bȍrü: Yak. börö (EDAL, 344); *ыт* собака от PTurk. *ɨt / *it: Yak. ɨt [18]; *хаас* гусь от PTurk. *Kāŕ: Yak. Xās [18]; *куба* лебедь от PTurk. *Kugu: Yak. kuba [18]; *сордоҥ* щука от Pturk. Čortan [19].

В отобранном материале встречаются также прототунгусские основы в некоторых названиях животных, вошедших в якутский язык из эвенкийского и эвенского языков: *моҕотой* бурундук от PTung. *moKo(lV)- (1 летучая мышь 2 бурундук): Evk. mokoločī 1; Evn. mokotoj 2. [18; *анды* турпан от PTung. *andi scoter, a k. of duck (турпан, утка-чернеть): Evk. anni, andi, ende [18].

В исследовании Н.В. Турантаевой и Н.В. Малышевой рассматривается 13 названий животных монгольского происхождения в современном якутском языке [20]. Из них в составе исследуемых гидронимов чаще встречаются: *кырса* песец от PMong. степная лиса [18]; *сиэгэн* россомаха от *jege-gen / jegegen, jegeken*, халх. *зээхэн*, калм. *зеекн*, бур. *зээгэн* (зоол.) ‘росомаха’ [21].

Таким образом, этимология зоокомпонентов гидронимов указывает на наличие трех основных стратов, лежащих в основе названий животных в якутском языке: тюркский, тунгусо-маньчжурский и монгольский.

Многовековые языковые контакты приводят к возникновению гибридов. Гибридные имена собственные, состоящие из лексических и морфологических элементов двух и более языков [4] встречаются во всех видах топонимов. Например, в гидронимах название озер Краскалах/ *Кырааскалаах* ‘с краской’, Улахан-Краскалах/ *Улахан Кырааскалаах* ‘большое с краской’, Аччыгый-Кырааскалаах/ *Аччыгый-Кырааскалаах* ‘малое с краской’ содержат гибридный топоним, имеющий в своей основе русское слова *краска*. В справочнике охотника на В.Н.Сивцева находим, что *кырааска* (лат. *Salvelinus czerskii*) (см. рис. 2) это голец Черского, рыба с красноватым мясом, с розовыми пятнышками в период размножения, с красными плавниками, попадающая из тундровых озер в нижнее течение рек Оленек, Лена, Яна, Индигирка, Алазея, Колыма [22]. Якутское название *диэрбэн* употребляется меньше, чем гибридное, образованное от русс. *краска* + аффикс обладания *-лаах*, букв. *‘с краской’*. Основанием для появления такого названия могла послужить яркая окраска рыбы, в отличие от других представителей ихтиофауны Якутии. Наличие таких гибридов является свидетельством длительных тесных языковых контактов.

Топонимическая стратиграфия территории Республики Саха (Якутия) обнаруживает многослойные страты, которые отражаются в топонимии, в том числе, в названиях водных объектов, этимология некоторых из них остается неясной до сих пор. Прототюркские, протомонгольские основы, слова, происходящие из тунгусо-маньчжурских (эвенкийского,

эвенского), юкагирского, чукотского языков требуют тщательного анализа и скрупулезной работы с лексикографическими источниками.

Рис. 2. Кырааска – якутское название гольца Черского (лат. Salvelinus czerskii).
Fig. 2. Kyraaska is the Yakut name of Chersky char (lat. Salvelinus czerskii).

Вопросы мотивированности топонимов возникают, когда речь идет о причинах, побудивших номинаторов назвать данный объект именно таким образом. Под мотивационным признаком в работе понимается вслед за А.В. Суперанской некоторое свойство предмета, положенное в основу названия [13]. Многие исследователи сходятся во мнении, что все разнообразие мотивационных признаков в семантике топонимов можно условно поделить на относящиеся к миру природы и к миру человека [23]. Данное наблюдение подтверждается и в отобранном корпусе гидронимов с зоокомпонентом, несмотря на то, что все они относятся, бесспорно, к миру природы, но именно человек дает название географическому объекту с учетом многих факторов, которые связывают его с животным миром, в первую очередь, в связи с его хозяйственной деятельностью.

Названия водных объектов содержат мотивационный признак “Рыбалка” (28%). В водоемах Якутии водится 55 морских и 45 пресноводных (пресноводных, полупроходных и проходных) видов рыб [24]. В составе 267 гидронимов встретился 31 ихтионим, название рыбы. Из них самым частотным является *собо* ‘карась’ (57 гидронимов), самая распространенная и почитаемая якутами за вкусовые качества рыба: оз. Собо (‘карась’), оз. Соболох/ *Соболоох* (‘с карасем’), оз. Алта-Собо (‘шесть карасей’), оз. Соболур/ *Соболуур* (‘где можно ловить карася’) и т.д. В семантике гидронима с ихтионимом есть прямое указание на то, что данный вид водится здесь в изобилии: оз. Сордоннох/ *Сордоҥноох* (‘с щукой’), оз. Сордоннур/ *Сордоҥнуур* (‘где можно ловить щуку’), оз. Сыаганнах/ *Сыаҕаннаах* (‘с налимом)’, Алысардах/ *Алыһардаах* (‘с окунем’), оз. Мундулах/ *Мундулаах* (‘с гольяном озерным’), Мундулур/ *Мундулуур* (где ‘ловят гольяна озерного’), Кысыл-Балыктах/ *Кыһыл Балыктаах* (‘с горбушей’), порой с указанием на разные качества рыбы, типа *Көтөх Суокурдаах* (‘с тощим колымским сигом’), *Сыа Сордоҥ* (‘жирная щука’).

Охота является испокон веков основным источником пропитания для народов Сибири и Дальнего Востока. Мотивационный признак “Охота” (27%) присутствует в гидронимах, содержащих названия промысловых животных, типа: река Кыл-Ытыалабыт/ *Кыыл Ытыалаабыт* (где ‘стреляли в дикого оленя/ лося’), речка Кис-Юрях/ *Киис Урэх* (‘соболь речка’) и др. В семантике 22 гидронимов содержится указание на постоянное проживание на территории водоема какого-либо вида: Кырса-Уялах/ *Кырса Уйалаах* (где ‘есть нора песца’), Эсе-Аргахтах/ *Эһэ Арҕахтаах* (‘с берлогой медведя’), река Бёрё-Хороно/ *Бөрө Хорооно* (‘нора волка’). Мотивационный признак «Охота» присутствует также в названиях водных объектов, содержащих зоокомпонент-орнитоним, то есть название птицы. Охота на водоплавающую дичь является одним из самых популярных видов сезонной охоты в Якутии. Всего в гидронимикон вошло 321 название с компонентом-орнитонимом, где самым популярным

является зоокомпонент *хаас* ‘гусь’, встретившийся 61 раз, как в наиболее распространенных в якутской топонимии конструкциях, типа Хас-Кюель/ *Хаас Күөл* (‘гусь озеро’), Хастыр/ *Хаастыыр* (‘где охотятся на гуся’), Хастах/ *Хаастаах* (‘где водится гусь’), так и в более семантически осложненных гидронимах, с событийной семантикой, уточняющей способ охоты, как в названии оз. Хас-Аялабыт/ *Хаас Айалаабыт* (где ‘охотились на гуся с луком-самострелом’) или передающих какой-то факт, случившийся на охоте: оз. Хас-Олорбута/ *Хаас Олорбута* (‘где сидел гусь’). К этому же типу относятся названия озер Кус-Кюеле/ *Кус Күөлэ* (‘озеро утки’), оз. Кустах/ *Кустаах* (‘с уткой’). Другие орнитонимы, встречающиеся в составе гидронимов, не относятся к промысловым видам, мотивационным признаком в данном виде гидронимов является «Распространенный вид в данной местности» (16%): оз. Турахтах/ *Тураахтаах* (‘с вороной’), оз. Хоптолох/ *Хоптолоох* (‘с чайкой’), оз. Суор-Уялах/ *Суор Уйалаах* (‘где есть гнездо ворона’), речка Куба Юрюете/ *Куба Үрүйэтэ* (‘ручей лебедя’). Некоторые виды занесены сейчас в Крансую книгу, но название сохранило память о тех временах, когда люди охотились на них: оз. Туруя-Тусахтабыт/ *Туруйа Туһахтаабыт* (‘где ставили силки на журавля’), оз. Кыталык-Сиебит/ *Кыталык Сиэбит* (‘съели стерха (белого журавля)’). Некоторые птицы являются тотемными животными, сакральными для народов Якутии, например тот же *кыталык* ‘стерх’ – это птица счастья в якутской культуре, считается, что тот, кто увидит танец белых журавлей, будет счастлив всю жизнь. Возникновение гидронима может быть связано с фактом нарушения табу на охоту на эти виды. В гидронимах встречаются случаи, когда название было дано по сходству очертаний озера с объектами живой природы, являющих собой пространственные метафоры (мотивационный признак «Форма водоема» (0,4%): оз. Хас-Сымыт/ *Хаас Сымыыт* (‘гусь яйцо’) или оз. Хас-Курдук/ *Хаас Курдук* (‘как гусь’) (см. рис. 3 и 4).

Якуты – самые северные коневоды, занимающиеся разведением лошадей и крупного рогатого скота в экстремальных климатических условиях. Мотивационный признак “Скотоводство” обнаружен в 262 гидронимах (26%), в них содержится порой прямое указание на то, что водоем и его берега благоприятны для разведения скота: оз. Ат-Ытар/ *Ат Ыытар* (‘где отпускают лошадь’), оз. Ат-Улур/ *Ат Уулуур* (‘где пьет/поят лошадь’), река Ынах-Юрюете/ *Ынах Үрүйэтэ* (‘коровий ручей’). Дробные системы деления по возрасту и полу домашнего скота являются свидетельством древних традиций северных скотоводов: оз. Аччыгый-Торбос (‘малое (озеро) теленок’ от *торбос* ‘теленок в первое лето после отела’), оз. Кулун-Кюель/ *Кулун Күөл* (‘жеребенок озеро’ от *кулун* ‘жеребенок с рождения и до шести-семи месяцев’), оз. Соногос/ *Соноҕос* (‘молодой, только что объезженный конь’) [15], река Убаса-Юрюе/ *Убаһа Үрүйэ* (‘ручей жеребёнка (с первой осени до года)’) и др.

Особое место занимает группа гидронимов, в которой встречаются компоненты-соматизмы, обозначающие части тела животного. Условно мы обозначили мотивационный признак в таких гидронимах “Череп животного как указатель” (3,6%). Так, в разных улусах (районах) Якутии и соседних регионах часто встречается компонент *муос* ‘рога’ (11 ед.) и *бас* ‘голова’ (41 ед.): названия Кыл-Муостах/ *Кыыл Муостаах* (‘с рогами дикого оленя/ лося’), Кыл-Бастаах/ *Кыл-Бастаах* (‘с головой дикого оленя/ лося’), Эсэ-Бастах/ *Эһэ Бастаах* (‘с головой медведя’). Традиция вешать на видное место черепа животных основывалась на необходимости делать заметные ориентиры при освоении территории. Встречаются также названия домашних животных в составе 31 гидронима с соматизмом *бас*: Ат-Бастах/ *Ат-Бастаах* (‘с головой лошади’), Сылгы-Бастах/ *Сылгы Бастаах* (‘с головой лошади’) и др. Помимо использования черепа животного в качестве ориентира, известно, что это может быть также свидетельством шаманских обрядов в былые времена. Домашний скот приносили в жертву злым верхним и нижним духам в целях исцеления от болезней [25]. При этом шкуру жертвенного животного вместе с головой размещали в лесу, подальше от жилья, закрепив ее специальным способом, подробно описанным в экспедиционных записях Г.Ф. Миллера в 1737 г. [26]. Со временем оставался только череп животного, не поддающийся тлену, и именно по такому черепу было названо много местностей.

Рис. 3. Озеро Хас-Сымыт, Быянгнырский наслег, Аллаиховский улус (район), Республика Саха (Якутия), геогр.координаты 70.874140, 147.753721 (снимок и данные из Яндекс Карты).
Fig. 3. Lake Khas-Symyt, Byyangnyrsky nasleg, Allaikhovsky ulus (district), Republic of Sakha (Yakutia), geographical coordinates 70.874140, 147.753721 (snapshot and data from Yandex Maps).

Рис. 4. Озеро Хас-Курдук, озеро Хас-Курдук, Олеринский Суктул, Нижнеколымский район, Республика Саха (Якутия), геогр.координаты 70.517683, 153.316188 (снимок и данные из Яндекс Карты).
Fig. 4. Lake Khas-Kurduk, Lake Khas-Kurduk, Olerinsky Suktul, Nizhnekolymsky district, Republic of Sakha (Yakutia), geographical coordinates 70.517683, 153.316188 (snapshot and data from Yandex Maps).

Таким образом, выявлены следующие основные мотивационные признаки в семантике гидронимов с зоокомпонентом Якутии: “Рыбалка” (28%), “Охота” (27%), “Скотоводство” (26%), «Распространенный вид в данной местности» (16%), “Череп-указатель” (3,6%), “Форма водоема” (0,4%).

3. Создание словарей и карт на платформе ЛингвоДок

Платформа ЛингвоДок, созданная для документирования исчезающих языков, позволяет создавать многослойные словари, с возможностью совместного пополнения словарных данных с сохранением истории действий пользователей, продвинутого поиска данных в словарях по множеству параметров [27]. Особо ценной для словаря топонимов является возможность поиска данных на карте с автоматическим построением топонимических изоглосс. Под топонимической изоглоссой (топоизоглоссой) понимается линия, нанесенная на топонимическую карту, показывающая территориальное распространение того или иного топонимического явления [4].

Для загрузки на платформу ЛингвоДок созданного словаря, необходимо сохранить его в формате CSV и пройти по инструкции все этапы, которые подробно описаны на платформе. Словарь можно «создать с нуля или в виде импортированного словаря из формата ранних версий ЛингвоДок и CSV со специальным символом-разделителем. CSV это текстовый формат представления табличных данных, его можно создать из файлов Excel, используя диалоговое окно экспорта Starling или любым другим способом» [28]. Автор словаря может дать доступ нескольким пользователям системы для совместной работы над словарем. Загруженный на платформу словарь можно редактировать дальше, вносить изменения, дополнения в любую колонку.

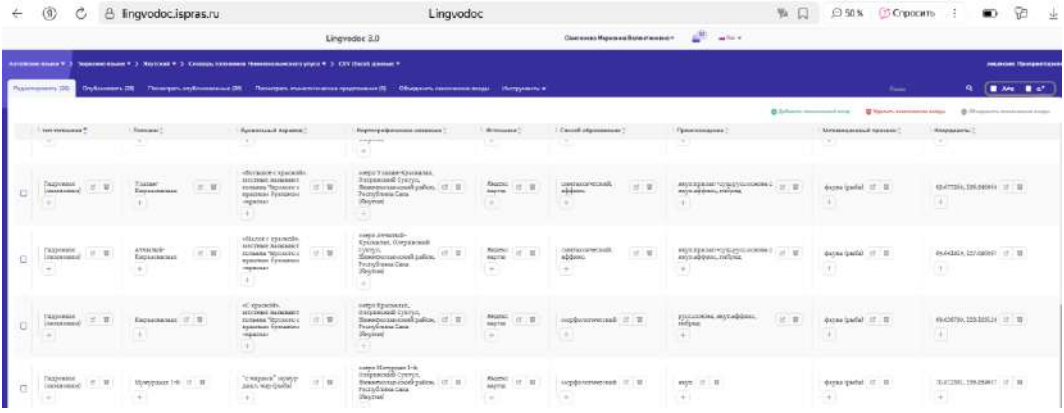


Рис. 5. Вид словаря гидронимов в режиме редактирования.
Fig. 5. View of the dictionary of hydronyms in editing mode.

После загрузки словаря на платформу, можно построить карту по любому критерию, по любому столбцу. Для построения точной карты требуются координаты из сервиса Яндекс Карты, в примере на рис. 5 они указаны в крайней колонке справа.

Картографирование данных выполняет поисковые запросы любого уровня сложности на выбранной территории, указанной в координатах. Распространение языковых явлений можно отметить на карте не только точками, но также изоглоссами, географическими ареалами, которые будут выделены выбранным цветом. Результаты картографирования возможно скачать в формате Excell. Построенная карта может быть сохранена в виде ссылки. При добавлении новых материалов на ЛингвоДок возможно ее автоматическое пополнение.

На карте ниже (рис. 6) представлены результаты поискового запроса построения карты, чтобы увидеть распространение гидронимов с зоокомпонентом *бөрө* ‘волк’. Хищник, представлявший во все времена угрозу для скота и лошадей, прочно вошел в названия местностей, как предупреждение: Бёрёлёх/ *Бөрөлөөх* (‘с волком/где есть волк”), Бёрё-Уялах/ *Бөрө Үйаалаах* (‘с норой волка’) и т.д. Но большая концентрация таких названий в центральной Якутии, свидетельствует не только о распространенности данного вида, но и этническом составе центральных улусов, где проживает больше якутского населения.

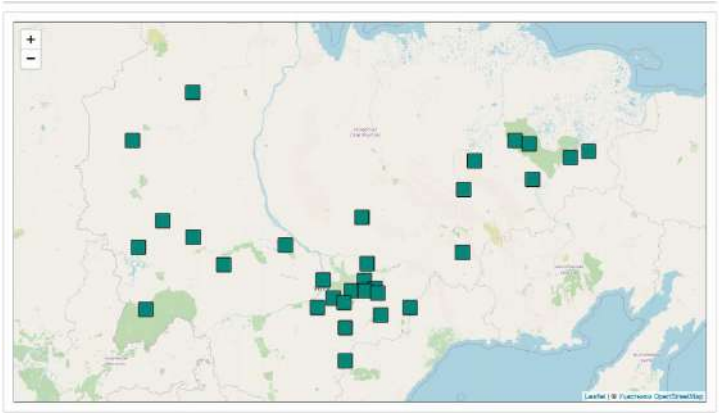


Рис. 6. Карта распространения гидронимов с зоокомпонентом бөрө 'волк'.
Fig. 6. Map of the distribution of hydronyms with the zoo component börö 'wolf'.

Подобные карты можно строить по любому из вышеописанных критериев, будь-то по морфологическому, структурному, этимологическому или семантическому признаку. Интересно продемонстрировать возможности карт ЛингвоДок в исследованиях происхождения топонимов. Топонимическая стратиграфия Республики Саха (Якутия) включает в себя несколько слоев: тюркские, монгольские основы в якутских словах, слова, происходящие из тунгусо-маньчжурских (эвенкийского, эвенского), юкагирского и чукотского языков. На картах ниже (рис. 7) представлены карты с топоизоглоссами распространения гидронимов по происхождению. В отобранном материале якутских гидронимов оказалось 65%, эвенкийских (17%), эвенских (12%), чукотских (3%), юкагирских (3%).

Рис. 7. Ареалы распространения гидронимов РС(Я) по происхождению.
Fig. 7. Areas of distribution of hydronyms of the Republic of Sakha (Yakutia) by origin.

Опция построения карт на платформе ЛингвоДок позволяет совместить данные нескольких карт. На рис. 8 мы видим на карте совмещенные результаты поиска в виде точек четырех цветов, а на рис. 9 данные по четырем языкам наложены друг на друга и отображены на одной карте, с 4-мя топоизоглоссами, иллюстрирующими ареалы распространения гидронимов на четырех языках.

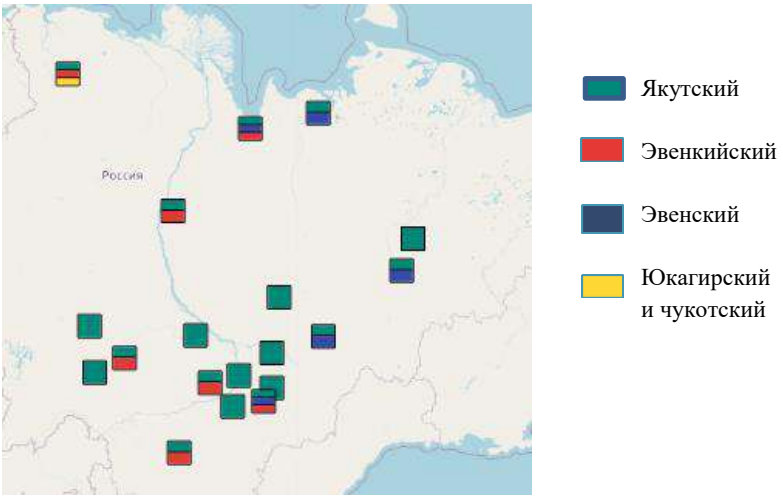


Рис. 8 Карта гидронимов Республики Саха (Якутия) по происхождению с нанесением точек на платформе ЛингвоДок.
Fig. 8 Map of hydronyms of the Republic of Sakha (Yakutia) by origin with points marked on the LingvoDoc platform.

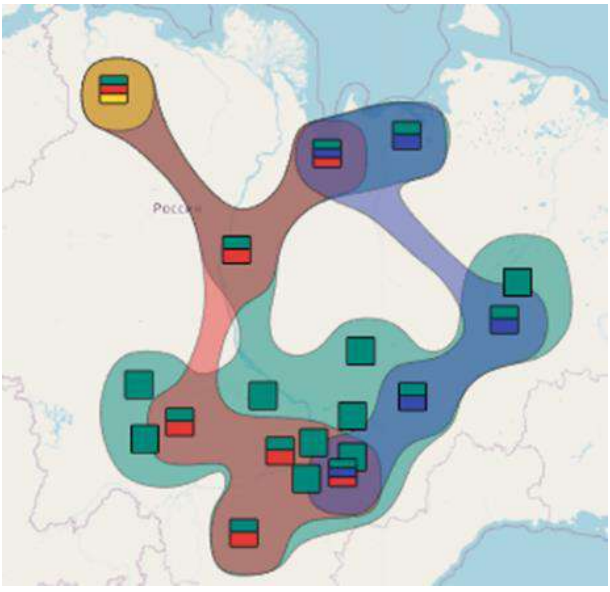


Рис. 9 Карта гидронимов Республики Саха (Якутия) по происхождению с нанесением точек и ареалов распространения на платформе ЛингвоДок.
Fig. 9 Map of hydronyms of the Republic of Sakha (Yakutia) by origin with points and areas of distribution marked on the LingvoDoc platform.

Карты позволяют также проследить различные языковые явления в масштабах одного района (улуса). 34 района республики имеют разную историю, этнический состав, языковые контакты, рельеф и т.д., что несомненно отражается в топонимии района. Рассмотрим, к примеру Горный улус, расположенный в центральной Якутии.

Этимологический анализ компонентов гидронимов показал, что из 47 гидронимов Горного улуса (рис. 10 и 11), 39 гидронимов содержат зоокомпонент якутского происхождения и 8 гидронимов содержат зоокомпонент эвенкийского происхождения. В настоящее время в

данном улусе проживает, по данным переписей населения 2010 г. и 2021 г. в основном якутское и русское население. Следует отметить, что в настоящее время языковая ситуация эвенкийского языка критическая [29], он практически вышел из употребления и сохраняется только в компактных местах проживания эвенков, к которым Горный улус не относится. Эвенкийский страт в топонимике свидетельствует об историческом прошлом района, о том, что раньше здесь жили эвенки. Данные топонимии могут указывать на места расселения предшествующих народов на данной территории, пролить свет на особенности уклада их жизни, отраженные в топонимии.

Рис. 10 Карта Горного улуса из Яндекс Карты.
Fig. 10 Map of Gorny Ulus from Yandex Maps.

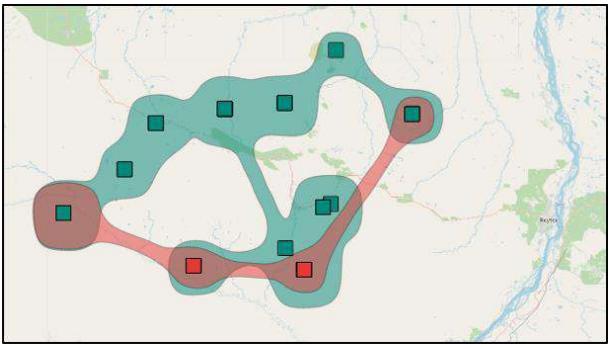


Рис. 11 Карта гидронимов Горного улуса по происхождению: якутские (зеленым цветом), эвенкийские (красным цветом).
Fig. 11 Map of hydronyms of the Gorny ulus by origin: Yakut (green), Evenki (red).

4. Заключение

В эпоху стремительного развития цифровых технологий инструменты платформы ЛингвоДок позволяют исследователям применять в анализе языкового материала опции для быстрой обработки данных на языках России. На примере словаря гидронимов Республики Саха (Якутия) продемонстрированы основные возможности платформы для структурирования объемного материала из 1000 названий водных объектов Якутии. Многоаспектный словарь, учитывающий в том числе реальное расположение на карте географических объектов, позволяет строить карты с изоглоссами распространения топонимических явлений на основе загруженных словарей топонимов с координатами. Выбранный тип гидронимов с зоокомпонентом вызывает интерес своим разнообразием как с морфологической, структурной, этимологической, так и семантической точек зрения. Комплексный подход к изучению топонимии большой территории позволяет прийти к

интересным выводам и находкам, в том числе относительно существования типа гидронимов, присущих только для данной территории.

Словарь будет дополняться и далее, платформа ЛингвоДок позволяет редактировать загруженные словари, возможно подключить несколько пользователей к работе над одним словарем. Опция построения карт в ЛингвоДок позволяет сделать интересные выводы по заселению территории, миграции народов, языковым контактам, культурным и языковым особенностям топонимической стратиграфии изучаемой территории.

Список литературы / References

[1]. Платформа LingvoDoc, Институт языкознания Российской академии наук, Институт системного программирования им. В.П. Иванникова Российской академии наук. Доступно по ссылке: https://lingvodoc.ispras.ru/, обращение 07.10.2025.

[2]. https://yandex.ru/maps

[3]. Гидрология. Сайт Федерального государственного бюджетного учреждения «Якутское управление по гидрометеорологии и мониторингу окружающей среды». URL: https://ykuthydromet.ru/product/gidrologiya/, дата обращения: 21.09.2025. // Hydrology. Website of the Federal State Budgetary Institution "Yakutsk Administration for Hydrometeorology and Environmental Monitoring". URL: https://ykuthydromet.ru/product/gidrologiya/, accessed: 21.09.2025.

[4]. Подольская Н. В. Словарь русской ономастической терминологии. Отв. ред. А. В. Суперанская. 2-е изд., перераб. и доп. М., Наука, 1988, 192 с. ISBN 5-256-00317-8. // Podolskaya, N. V. Dictionary of Russian Onomastic Terminology. Ed. A. V. Superanskaya. 2nd ed., revised and enlarged. Moscow, Nauka, 1988, 192 p. ISBN 5-256-00317-8.

[5]. Агеева Р.А. Происхождение имён рек и озёр. М., Наука, 1985, 146 с. // Ageeva R.A. Origin of names of rivers and lakes. M., Nauka, 1985, 146 p.

[6]. Мурзаев Э. М. Словарь народных географических терминов. М., Мысль, 1984, 656 с. // Murzaev E. M. Dictionary of folk geographical terms. M., Mysl, 1984, 656 p.

[7]. Багдарыын Сюлбэ. Топонимика Якутии: крат. науч.-попул. очерк. [отв. ред. П. А. Слепцов]; Акад. наук Респ. Саха (Якутия), Ин-т гуманит. исслед. Якутск, Бичик, 2004, 190 с. // Bagdaryyn Syulbe. Toponymy of Yakutia: brief. popular science essay, [rep. ed. P. A. Sleptsov]; Academician Sciences Rep. Sakha (Yakutia), Institute of Humanities. research Yakutsk, Bichik, 2004, 190 p.

[8]. Багдарыын Сүлбэ. Талыллыбыт үлэлэр (Избранные труды) – Дьокуускай, Бичик, 2013. Т. 5: Нөрүөн нөргүй буолуохтун! (Вечная память им!). 2016, 702 с. (на якутском языке) // Bagdaryyn Sulbe. Talylybyt yleler (Selected works). Yakutsk, Bichik, 2013. T. 5: Nөrүөn nөrgy buoluohtun! (Eternal memory to them!). 2016, 702 p. (in Yakut language).

[9]. Багдарыын Сүлбэ. Мэнэ ааттар. Якутскай, Кинигэ изд-вота, 1979, 288 с. (Багдарын Сюлбэ. Вечные названия. Якутск, Книжное издательство, 1979, 288 с.) // Bagdaryyn Sulbe. Меңе ааттar. Yakutsk, Kinige publishing house, 1979, 288 p. (Bagdaryn Syulbe. Eternal names. Yakutsk, Book publishing house, 1979, 288 p.).

[10]. Иванов Н.М. Функционирование топонимов в республике саха (Якутия). Вестник СВФУ. 2014. №1. URL: https://cyberleninka.ru/article/n/funktsionirovanie-toponimov-v-respublike-saha-yakutiya (дата обращения: 28.09.2025) // Ivanov N.M. Functioning of toponyms in the Sakha Republic (Yakutia). Bulletin of NEFU. 2014. no. 1. URL: https://cyberleninka.ru/article/n/funktsionirovanie-toponimov-v-respublike-saha-yakutiya (date of access: 09/28/2025).

[11]. Слепцов Ю.А. Гидронимы эвенов Момского района Якутии (из полевых материалов) // Культура и цивилизация. 2019. Том 9. № 4А. С. 76-83/ Sleptsov Yu.A. Hydronyms of the Evens of the Momsky region of Yakutia (from field materials) // Culture and civilization. 2019. Volume 9. No. 4A. pp. 76-83.

[12]. Курилов Ю.Г. Лексико-семантическая и структурно-морфологическая характеристика юкогирских топонимов. Известия РГПУ им. А. И. Герцена. 2008. №80. URL: https://cyberleninka.ru/article/n/leksiko-semanticheskaya-i-strukturno-morfologicheskaya-harakteristika-yukogirskih-toponimov (дата обращения: 28.09.2025) // Kurilov Yu.G. Lexical-semantic and structural-morphological characteristics of Yukogir toponyms. Bulletin of the A.I. Herzen State Pedagogical Univ. 2008. no. 80. URL: https://cyberleninka.ru/article/n/leksiko-semanticheskaya-i-strukturno-morfologicheskaya-harakteristika-yukogirskih-toponimov (date of access: 09/28/2025).

[13]. Суперанская, А.В. Структура имени собственного. Фонология и морфология. Москва, Наука, 1969, 206 с. // Superanskaya, A.V. The structure of a proper name. Phonology and morphology. Moscow, Nauka, 1969, 206 p.

[14]. Донидзе Г.И. Инструкция по русской передаче географических названий Якутской АССР. Москва, Издательство «Наука», Главная редакция восточной литературы, 1976, 37 с. // Donidze G.I. Instructions for the Russian transmission of geographical names of the Yakut ASSR. Moscow, Nauka Publishing House, Main Editorial Board of Eastern Literature, 1976, 37 p.

[15]. Большой толковый словарь якутского языка = Саха тылын быһаарыылаах улахан тылдьыта / Академия наук Республики Саха (Якутия), Институт гуманитарных исследований; под общей редакцией П. А. Слепцова. Новосибирск, Наука, 2004-2018. В 15-ти томах. // Large explanatory dictionary of the Yakut language = Sakha tylyn byhaaryylaakh ulakhan tyldyta / Academy of Sciences of the Republic of Sakha (Yakutia), Institute of Humanitarian Research; under the general editorship of P. A. Sleptsov. Novosibirsk, Science, 2004-2018. In 15 volumes.

[16]. Пекарский Э.К. Словарь якутского языка в 3-х томах. Акад. наук СССР. 2-е изд., М., АН СССР, 1958-1959. т. 1, вып. 1-4, 1958, 1279 с. // Pekarsky E.K. Dictionary of the Yakut language in 3 volumes, M., Academician Sciences of the USSR. 2nd ed., M., USSR Academy of Sciences, 1958-1959. vol. 1, no. 1-4, 1958, 1279 pp.

[17]. Грамматика современного якутского литературного языка : [фонетика и морфология] / Л.Н. Харитонов и др.; ред. Е.И. Убрятова ; АН СССР, Сиб. отд-ние Якут. фил.. ин-т яз., лит. и истории М., Наука, 1982, 496 с., с. 34-35 // Grammar of the modern Yakut literary language: [phonetics and morphology] / L.N. Kharitonov and others; ed. E.I. Ubryatova; Academy of Sciences of the USSR, Sib. Yakut department. Phil. Institute of Languages., Lit. and history M., Nauka, 1982, 496 p., pp. 34-35.

[18]. S.A. Starostin, A.V. Dybo, O.A. Mudrak. An Etymological Dictionary of Altaic Languages. 3 vol., Leiden; Boston, 2003, 1559 p.

[19]. Сравнительно-историческая грамматика тюркских языков. Региональные реконструкции. Отв.ред. Э.Р. Тенишев. М., Наука, 2002, 767 с. // Comparative-historical grammar of the Turkic languages. Regional reconstructions / Ed. E.R. Tenishev. M., Nauka, 2002, 767 p.

[20]. Турантаева Н.В., Малышева Н.В. К вопросу о монгольских заимствованиях в якутской лексике (на материале названий пушных животных). Вестник СВФУ. 2024. №3 (25). URL: https://cyberleninka.ru/article/n/k-voprosu-o-mongolskih-zaimstvovaniyah-v-yakutskoy-leksike-na-materiale-nazvaniy-pushnyh-zhivotnyh (дата обращения: 19.09.2025) // Turantaeva N.V., Malysheva N.V. On the issue of Mongolian borrowings in the Yakut vocabulary (based on the names of fur animals). Bulletin of NEFU, 2024, no. 3 (25). URL: https://cyberleninka.ru/article/n/k-voprosu-o-mongolskih-zaimstvovaniyah-v-yakutskoy-leksike-na-materiale-nazvaniy-pushnyh-zhivotnyh (access date: 09.19.2025).

[21]. Этимологический словарь монгольских языков: в 3 тт. Отв. ред. Г.Д. Санжеев, ред.-сост. Л.Р. Концевич, В.И. Рассадин, Я.Д. Леман. Институт востоковедения РАН. М., ИВ РАН, 2018, 232 с. // Etymological dictionary of Mongolian languages: in 3 volumes. Ed. G.D. Sanjeev, compiled by L.R. Kontsevich, V.I. Rassadin, Ya.D. Lehman. Institute of Oriental Studies of the Russian Academy of Sciences. Moscow, Institute of Oriental Studies of the Russian Academy of Sciences, 2018, 232 p.

[22]. Сивцев В.Н. Булчут кинигэтэ (Книга охотника). Дь., Бичик, 2017, 124 с. // Sivtsev V.N. Bulchut kinigete (Book of the Hunter). Yakutsk, Bichik, 2017, 124 p.

[23]. Доржиева Г.С. Топонимия Квебека как отражение франкоязычной культуры региона: этнолингвистический аспект. Диссертация на соискание ученой степени доктора филологических наук. Москва, 2012, 591 с. // Dorzhieva G.S. Toponymy of Quebec as a reflection of the French-speaking culture of the region: ethnolinguistic aspect. Dissertation for the degree of Doctor of Philological Sciences. Moscow, 2012, 591 p.

[24]. Кириллов А.Ф., Черешнев И.А. Аннотированный список рыбообразных и рыб морских и пресных вод Якутии. Вестник СВФУ. 2006. №4. URL: https://cyberleninka.ru/article/n/annotirovannyy-spisok-ryboobraznyh-i-ryb-morskih-i-presnyh-vod-yakutii (дата обращения: 23.09.2025) // Kirillov A.F., Chereshnev I.A. Annotated list of fishes and fishes of marine and fresh waters of Yakutia // Bulletin of NEFU, 2006, no. 4. URL: https://cyberleninka.ru/article/n/annotirovannyy-spisok-ryboobraznyh-i-ryb-morskih-i-presnyh-vod-yakutii (date of access: 09/23/2025).

[25]. Костырко В.С. Скот злых духов и жертвоприношения в шаманских обрядах. Вестник РГГУ. Серия: Литературоведение. Языкознание. Культурология. 2007. №7. URL: https://cyberleninka.ru/article/n/skot-zlyh-duhov-i-zhertvoprinosheniya-v-shamanskih-obryadah-1 (дата обращения: 22.09.2025) // Kostyrko V.S. Cattle of evil spirits and sacrifices in shamanic rites. Bulletin

of the Russian State University for the Humanities. Series: Literary criticism. Linguistics. Cultural studies. 2007, no. 7. URL: https://cyberleninka.ru/article/n/skot-zlyh-duhov-i-zhertvoprinosheniya-v-shamanskih-obryadah-1 (date of access: 09.22.2025).

[26]. Языческие верования и обряды. Описание якутского обряда жертвоприношения. По следам академического отряда Великой Северной экспедиции. Наука из первых рук, том 18, № 6, 2007. URL: https://scfh.ru/papers/opisanie-yakutskogo-obryada-zhertvoprinosheniya (дата обращения: 10.10.2025). // Pagan beliefs and rituals. Description of the Yakut sacrificial rite. In the footsteps of the academic detachment of the Great Northern Expedition. Science First Hand, vol. 18, no. 6, 2007. URL: https://scfh.ru/papers/opisanie-yakutskogo-obryada-zhertvoprinosheniya/ (date of access: 10.10.2025).

[27]. Lingvodoc: виртуальная лаборатория для документации исчезающих языков. URL: https://www.ispras.ru/technologies/lingvodoc/ (дата обращения: 21.09.2025) // Lingvodoc: a virtual laboratory for the documentation of endangered languages. URL: https://www.ispras.ru/technologies/lingvodoc/ (accessed: 21.09.2025).

[28]. Норманская Ю.В., Борисенко О.Д. Программная система Lingvodoc и возможности, которые она предлагает для документирования и анализа обско-угорских языков. Доклады Российской академии наук. Математика, информатика, процессы управления, 2022, Т. 504, № 1, стр. 60-82 // Normanskaya Yu.V., Borisenko O.D. The Lingvodoc software system and the possibilities it offers for documenting and analyzing Ob-Ugric languages. Reports of the Russian Academy of Sciences. Mathematics, informatics, control processes, 2022, vol. 504, no. 1, pp. 60-82.

[29]. «Языковое гнездо Эвэды»: Компания Нордголд профинансировала программу развития эвенкийского языка. URL: https://sakhapress.ru/archives/299365 (дата обращения: 22.09.2025) // "Eveda's Language Nest": Nordgold Company financed the Evenki language development program. URL: https://sakhapress.ru/archives/299365 (accessed: 22.09.2025).

Информация об авторах / Information about authors

Марианна Валентиновна САМСОНОВА – кандидат филологических наук, доцент, заведующая кафедрой французской филологии Института зарубежной филологии и регионоведения Северо-Восточного федерального университета. Сфера научных интересов: топонимика, гидронимика, семантика, стилистика, фразеология.

Marianna Valentinovna SAMSONOVA – Cand. Sci. (Philology), associate professor, and head of the French Philology Department at the Institute of Foreign Philology and Regional Studies at North-Eastern Federal University. Her research interests include toponymy, hydronymy, semantics, stylistics, and phraseology.

DOI: 10.15514/ISPRAS-2025-37(6)-27

Язык памятников прибалтийско-финской письменности XVII-XIX вв.: комплексный анализ на базе лингвистической платформы LingvoDoc (введение в проект)

С.В. Нагурная, ORCID: 0000-0002-6233-8045 <kov@krc.karelia.ru>

Институт языка, литературы и истории Карельского научного центра РАН, Россия, 185910, Республика Карелия, г. Петрозаводск, ул. Пушкинская, д. 11,

Аннотация. В статье дана характеристика проекта, реализация которого начата в текущем году в Институте языка, литературы и истории Карельского научного центра РАН, – «Язык памятников прибалтийско-финской письменности XVII-XIX вв.: комплексный анализ на базе лингвистической платформы LingvoDoc». Платформа LingvoDoc – цифровое хранилище, предназначенное для резервирования языковых данных, инструменты которого предоставляют возможность одновременно осуществлять обработку языкового материала и проводить в онлайн-режиме анализ фонетических, морфологических, лексических особенностей языка. Размещение текстов памятников карельской и вепсской письменностей на платформе LingvoDoc позволит не только решать исследовательские задачи (текстологический анализ, выявление диалектной специфики, создание конкордансов и т.д.), но и выйти на решение вопросов языковой документации. Метод обработки больших данных обеспечит релевантность результатов.

Ключевые слова: карельский язык; вепсский язык; платформа LingvoDoc; памятники письменности; анализ текста; документация языков.

Для цитирования: Нагурная С.В. Язык памятников прибалтийско-финской письменности XVII-XIX вв.: комплексный анализ на базе лингвистической платформы LingvoDoc (введение в проект). Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 169–176. DOI: 10.15514/ISPRAS–2025–37(6)–27.

Благодарности: Статья подготовлена за счет гранта Российского научного фонда No 25-78-20006. Результаты получены с использованием услуг Центра коллективного пользования Института системного программирования им. В.П. Иванникова РАН – ЦКП ИСП РАН.

The Language of Baltic-Finnic Literary Monuments of the 17th–19th Centuries: A Comprehensive Analysis Based on the LingvoDoc Linguistic Platform (Introduction to the Project)

S.V. Nagurnaja, ORCID: 0000-0002-6233-8045 <kov@krc.karelia.ru>

Institute of Linguistics, Literature and History of KarRC RAS,
11, Pushkinskaya st., Petrozavodsk, 185910, Russia.

Abstract. The article describes a project, the implementation of which began this year at the Institute of Language, Literature and History of the Karelian Scientific Centre of the Russian Academy of Sciences: "The Language of the Monuments of the Baltic-Finnic Literature of the 17th-19th Centuries: A Comprehensive Analysis Based on the LingvoDoc Linguistic Platform." The LingvoDoc platform is a digital repository designed to back up language data. Its tools allow for the simultaneous processing of language material and the online analysis of phonetic, morphological, and lexical features of the language. Placing texts from Karelian and Vepsian scripts on the LingvoDoc platform will not only enable research tasks (textual analysis, identifying dialectal specifics, creating concordances, etc.) but also address issues of language documentation. Big data processing will ensure the relevance of the results.

Keywords: Karelian language; Vepsian language; LingvoDoc platform; written monuments; text analysis; language documentation.

For citation: Nagurnaya S.V. Language of Baltic-Finnic written monuments of the 17th-19th centuries: a comprehensive analysis based on the LingvoDoc linguistic platform (introduction to the project). Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 169-176 (in Russian). DOI: 10.15514/ISPRAS-2025-37(6)-27.

Acknowledgements. The reported study was granted by Russian Science Foundation, project no. 25-78-20006. The results were obtained using the services of the Ivannikov Institute for System Programming (ISP RAS) Data Center.

1. Введение

В последние годы значительно возрос интерес исследователей к вопросам истории языка. Интерпретация лингвистического материала памятников позволяет реконструировать исторические варианты языков. Кроме того, для ряда языков и диалектов в ходе исследований уточняется «…релевантность диалектно-дифференцирующих изоглосс, на основе которых строятся их современные классификации» [1]. Не являются исключением и прибалтийско-финские языки Карелии, карельский и вепсский, письменные стандарты которых активно развиваются с начала 1990-х гг., но формирование их письменных традиций уходит в глубь веков.

Письменное наследие карельского и вепсского языков не отличается многообразием, поэтому каждый из имеющихся памятников письменности – это уникальный ресурс для изучения истории языков и народов. Памятники письменности – важнейшие документы не только исторической фонетики, грамматики, лексикологии и диалектологии языков, но и этногенеза говорящих на нем этнических групп. Каждый из них, независимо от жанра, является незаменимой деталью, запечатлевшей язык в определенный момент его развития.

В текущем году Российским научным фондом поддержан реализуемый в Институте языка, литературы и истории Карельского научного центра РАН проект «Язык памятников прибалтийско-финской письменности XVII-XIX вв.: комплексный анализ на базе лингвистической платформы LingvoDoc». В соответствии с конкурсной документацией, исследование осуществляется на базе существующей научной инфраструктуры мирового уровня, которую представляет собой система LingvoDoc [2], предназначенная «для составления, анализа и хранения словарей, корпусов и конкордансов языков и диалектов».

В числе целей предпринимаемого исследования:

1) текстологическое изучение материалов известных на сегодняшний день старокарельских и старовепсских рукописных и печатных памятников письменности, уточнение связей между ними;
2) целостное описание графо-фонетических, морфологических, лексических систем памятников;
3) определение (уточнение) диалектной специфики памятников;
4) введение в научный оборот нового лингвистического материала, что позволит представить историю карельского и вепсского языков как непрерывный процесс.

В соответствии с перечисленными целями одной из важнейших задач при реализации проекта представляется целенаправленная работа по выявлению, оцифровке, расшифровке и созданию на базе платформы LingvoDoc коллекции текстов разножанровых памятников карельской и вепсской письменностей. Инструментарий лингвистической платформы позволяет осуществлять расширенный фонологический и морфологический анализ текстов памятников, а также произвести диалектную локализацию языка ранних письменных источников. В результате выполнения указанных задач одновременно будет решаться одна из основных проблем современного филологического знания – цифровое сохранение миноритарных языков.

2. Обзор проекта

2.1 Предметная область исследования

Официально карельская и вепсская письменности получили развитие в 1930-е гг., однако ранние рукописные памятники фиксировались значительно раньше [3-5]. В силу своей уникальности и единичности каждый из них представляет собой особую ценность для исследователей.

К самым ранним прибалтийско-финским записям относятся берестяные грамоты, обнаруженные в Новгороде – грамота № 292 с записью заклинания от молнии (1240-1260 гг.), грамота № 403, содержащая русско-карельский словарик (1360-1380 гг.) и др. Используемый в берестяных грамотах язык мог быть отражением некоего северо-восточного прибалтийско-финского койне – языковой формы, объединившей черты нескольких прибалтийско-финских диалектов. Языковая форма зародилась на полиэтнической древненовгородской территории [6]. Грамоты содержат собственные личные имена и топонимы с карельскими элементами, а также лексику прибалтийско-финского, в том числе карельского, происхождения.

Фиксации карельских топонимов и антропонимов в изобилии встречаются в новгородских и шведских документах позднего Средневековья.

Основную часть ранних рукописных и печатных памятников XVII-XIX вв. представляют словарные записи, грамматические очерки в этнографических описаниях, переводы текстов духовного содержания и учебные пособия.

В XVII в. кириллицей были записаны уникальные тексты десяти карельских заговоров, в которых можно найти черты всех наречий карельского (людиковского, ливвиковского и собственно карельского) и вепсского языков, неоднородные по своим фонетико-морфологическим признакам. Язык этих текстов является предшественником карельского и вепсского языков, поскольку процесс формирования наречий карельского языка в XVII в. еще продолжался. Этот источник ждет дальнейшего исследования.

К XVII в. относится русско-карельская словарная запись из списков Азбучного патерика библиотеки Соловецкого монастыря. К настоящему времени установлено, что время создания записи – 1666-1668 годы [7]. Также определены языковые особенности, позволяющие верифицировать карельскую диалектную основу словарной записи и описать характерные признаки карельского языка XVII в.

Не так давно обнаружена ранее неизвестная карело-русская запись 1668 года в авторском сборнике древнерусского книжника Прохора Коломнятина. Запись содержит 600 лексем и отражает язык тверских карелов [8].

Ранние опубликованные лексические материалы на карельском языке в некоторой степени известны благодаря историко-филологическим исследованиям ученых-этнографов XVIII-XIX вв. В поисках древнего мирового языкового универсума ученые сопоставляли лексические формы разных языков. Результатом такой деятельности стал, в частности, словарь тезаурусного типа П. Палласа «Сравнительные словари всех языков и наречий», в который вошли карельские лексические материалы. В настоящее время определена диалектная специфика карельских материалов словаря [9-10]. Исследование осуществлялось с применением инструмента «Анализ когнатов» на лингвистической платформе LingvoDoc.

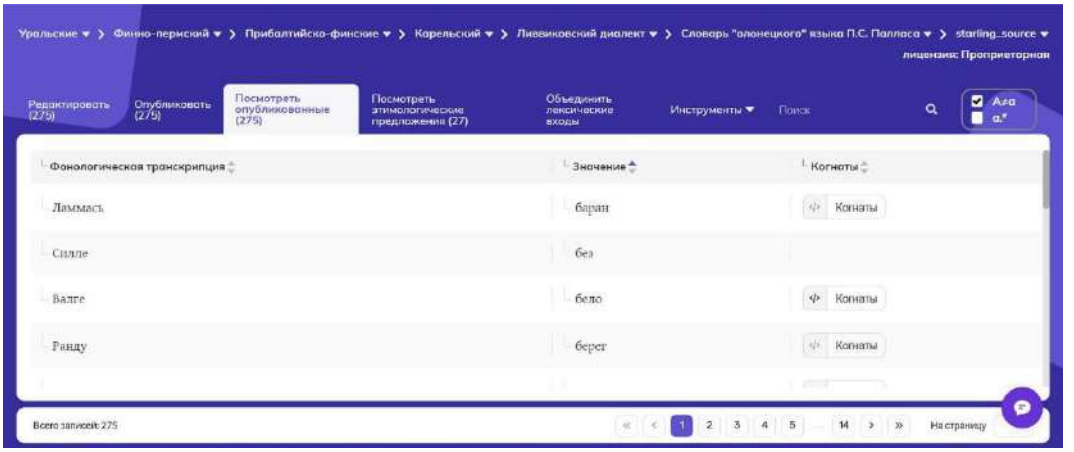


Рис. 1. Словарь «олонецкого» языка П.Палласа на платформе LingvoDoc.
Fig. 1. Dictionary of the "Olonets" language by P. Pallas on the LingvoDoc platform.

Практически все известные памятники карельской и вепсской письменностей были созданы на кириллической графике. Программы LingvoDoc позволяют обрабатывать тексты с сохранением графики, что имеет важное значение для описании памятников письменности.

Редким исключением являются, к примеру, записи члена-корреспондента Петербургской академии наук Федора Кеппена, который, доказывая родство индоевропейского и финно-угорского племен, в одной из своих работ в составе общей финно-угорской лексики привел ряд карельских и вепсских слов, зафиксированных латиницей.

Сведения о карелах и карельском языке содержались в этнографических описаниях Олонецкой и Архангельской губерний, некоторые из которых включали языковые материалы. Примером может служить произведение известного российского ученого-этнографа В.А. Дашкова «Описание Олонецкой губернии в историческом, статистическом и этнографическом отношениях» 1841 года, в заключительной части которого содержатся грамматический очерк карельского языка и небольшой карельско-русский словарь (чуть более 70 слов, записанных русскими буквами).

Значительная по объему и разнообразию карельская лексика представлена в труде А.В. Старчевского «Проводник и переводчик по отдаленнейшим окраинам России», опубликованном в 1889 г. в Санкт-Петербурге. Современные исследования карельского материала показали, что в работе использована разнодиалектная лексика [11].

Памятников вепсской письменности на сегодняшний день известно значительно меньше, чем карельской. Прежде всего, это указанные выше заговоры XVII в. Рукопись с записью заговоров обнаружил в Заонежье российский филолог В.И. Срезневский. Известный финский языковед Э. Сетяля, консультировавший В.И Срезневского, предположил, что нерусские

записи сделаны на вепсском языке. При последующей работе с документом ученые пришли к заключению, что записи являются сплавом нескольких близкородственных языков южной Карелии, и часть из них – это вепсско-людиковские заговоры, записанные кириллицей.

Второй, более поздний, памятник относится к середине XIX в. Он был обнаружен в делах канцелярии Олонецкого губернатора за 1848 г. и представляет собой рукописный русско-вепсский словарь, составленный вытегорским исправником, который назвал свою работу «Лексиконом корельского языка». Лингвистическое исследование памятника показало, что в нем представлен куштозерский говор вепсского языка, входящий в группу восточных, вологодских средневепсских говоров [12]. Поскольку куштозерские вепсы давно ассимилировались, данный «Лексикон» представляет несомненный интерес для вепсской исторической диалектологии.

Записи карельской и вепсской лексики встречались в неопубликованных трудах по финно-угроведению. В частности, в Архиве Российской академии наук в С.-Петербурге сохранилась рукопись многоязычного глоссария А.М. Шегрена, включающего 360 лексем с переводами на саамский, ненецкий, коми языки, а также на ливвиковское наречие карельского языка и вепсский язык. Рукопись отражает полевую работу Шегрена 1824-1829 гг., когда ученый находился в длительной экспедиции, побывав, в том числе, в Олонецкой и Северной Карелии.

В ходе реализации анонсируемого проекта нам удалось познакомиться с фондом А.М. Шегрена. В результате начатой обработки карельских и вепсских материалов часть из них уже помещена на платформу LingvoDoc (в частности, список русских слов с переводом на «чудский» язык из фонда № 94 и др.).

2.2 Значение исследования

Сравнительно-сопоставительное исследование вводимых текстов памятников письменностей карельского и вепсского языков на лингвистическую платформу LingvoDoc будет способствовать не только решению задач, стоящих перед сравнительно-историческим языкознанием (реконструкция исторических форм языков, исследование диахронических процессов и т.д.), но и решению одной из фундаментальных проблем современного гуманитарного знания – цифровой документации языков. В настоящее время документальная лингвистика является активно развивающейся областью прикладной лингвистики. Современные исследования в сфере документации языков направлены на обеспечение долгосрочных, всеобъемлющих и многоцелевых записей лингвистических практик, характерных для речевых сообществ. В числе таких лингвистических практик – памятники письменности.

Одна из важнейших целей документирования языка – предоставление данных для дальнейших теоретических и прикладных исследований языков, находящихся под угрозой исчезновения. Для языковой документации используются различные инструменты лингвистических технологий. Современный лингвистический исследовательский инструментарий, сосредоточенный на базе платформы LingvoDoc, позволяет посмотреть на уникальный материал памятников карельской и вепсской письменности под новым углом, найти ответы на вопросы исторической фонетики, грамматики, лексикологии и диалектологии, которые до настоящего времени оставались нерешенными.

Теоретическая значимость исследования обусловлена возможностью реконструкции грамматических систем карельского и вепсского языков анализируемого периода и определения хронологии основных языковых явлений. Важным представляется также систематизация принципов кириллической транскрипции, характерных для памятников письменности прибалтийско-финских языков данного периода.

Изучение памятников письменности имеет также важное прикладное значение для современной ситуации, поскольку формирование современной письменной традиции невозможно без учета истории карельского и вепсского языков. Анализ материалов

памятников разных жанров, разного временного и территориального охвата с применением современных методик позволяет отследить историю языковых явлений, определить основные направления их контактирования, выявить архаизмы. Изучение истории языка, истории становления письменности является отправной точкой в процессе создания нормированных языков. Выявление и исследование исторических форм языка может способствовать развитию и обогащению его определенных фонетических, морфологических и синтаксических свойств. Проект имеет выход в современную практику развития новописьменных вариантов карельского и вепсского языков, поскольку корпусом карельских и вепсских старописьменных текстов можно воспользоваться при составлении исторической грамматики, толкового, этимологического и диалектных словарей, в преподавании вузовских курсов (история языка, диалектология, лексикология, финно-угроведение, феннистика и т.д.).

Основными мероприятиями, осуществляемыми в рамках реализации проекта, таким образом, являются:

- инвентаризация и систематизация имеющихся в распоряжении лингвистов рукописных и печатных памятников карельского и вепсского языков, их текстологический анализ;
- выявление новых памятников письменности;
- оцифровка, расшифровка и аннотирование текстов;
- загрузка материалов на лингвистическую платформу LingvoDoc, создание конкордансов;
- анализ когнатов, определение диалектной принадлежности текстов памятников, описание фонологических особенностей;
- создание морфологических словарей памятников, выявление и описание морфологических особенностей текстов, их сопоставительный анализ;
- анализ лексического состава памятников письменности;
- разработка базы данных «Памятники письменности карельского и вепсского языков» и создание на ее основе электронного путеводителя.

Таким образом, системный анализ памятников карельской и вепсской письменностей позволит существенно расширить имеющиеся на сегодняшний день представления о развитии и становлении карельского и вепсского языков, осуществить реконструкцию языковых форм определенного исторического периода. Размещение текстов памятников на платформе LingvoDoc, в свою очередь, предоставит возможность использования современных методов анализа этих текстов и их цифрового документирования.

Список литературы / References

[1]. Кошелюк Н. А. Чусовой словарь из архива Г. Ф. Миллера: особенности и его значение для классификации мансийских диалектов // Oriental Studies, № 5, 2023 г., стр. 1309–1324. DOI: 10.22162/2619-0990-2023-69-5-1309-1324. / Koshelyuk N. A. Dictionary of Chusovaya Mansi from G. F. Müller's Archives: Its Features and Significance for Mansi Dialect Classification. Oriental Studies, 2023, No 5, pp. 1309–1324 (in Russian). DOI: 10.22162/2619-0990-2023-69-5-1309-1324.

[2]. Платформа LingvoDoc, Институт языкознания Российской академии наук, Институт системного программирования им. В.П. Иванникова Российской академии наук. Доступно по ссылке: https://lingvodoc.ispras.ru/, обращение 07.10.2025.

[3]. Нагурная С.В. Карельская письменность. Народы Карелии: историко-этнографические очерки. Петрозаводск, Periodika, 2019, стр. 65–77. / Nagurnaya S.V. Karelian script. Peoples of Karelia: historical and ethnographic essays. Petrozavodsk, Periodika, 2019, pp. 65-77 (in Russian).

[4]. Памятники лексикографии в «Цветнике» Прохора Коломнятина 1668 года. Исследования и тексты. М.; СПб., Альянс-Архео, 2023, 634 с. / Monuments of lexicography in Prokhor Kolomnyatin's

"Flowerbed" of 1668. Research and texts. Moscow, Saint Petersburg, Alliance-Archeo, 2023, 634 p. (in Russian).

[5]. Муллонен И. И., Новак И. П. Древнекарельские языковые особенности в памятниках письменности второй половины XVII — начала XVIII в. Вестник Санкт-Петербургского университета. Язык и литература, № 21 (3), 2024 г., стр. 720–740. DOI: 10.21638/spbu09.2024.312. / Mullonen I. I., Novak I. P. Old-Karelian language features in written language heritage of the second half of the 17th — early 18th centuries. Vestnik of Saint Petersburg University. Language and Literature, 2024, No 21 (3), pp. 720–740 (in Russian). DOI: 10.21638/spbu09.2024.312.

[6]. Laakso J. Vielä kerran itämerensuomen vanhimmista muistomerkeistä. Virittäjä, 4, 1999, pp. 531-555.

[7]. Муллонен И. И., Панченко О. В. Первый карельско-русский словарь и его автор афонский архимандрит Феофан. Петрозаводск, ПетрГУ, 2013, 116 с. / Mullonen I. I., Panchenko O. V. The first Karelian-Russian dictionary and its author, the Athonite archimandrite Feofan. Petrozavodsk, PetrSU, 2013, 116 p. (in Russian).

[8]. Савельева Н. В. Неизвестные памятники лексикографии в «Цветнике» Прохора Коломнятина. Русская литература, № 3. 2019 г., стр. 54–63. / Savelyeva N. V. Unknown monuments of lexicography in Prokhor Kolomnyatin's "Flowerbed". Russian literature, 2019, No 3, pp. 54–63 (in Russian).

[9]. Новак. И. П. Определение диалектной специфики карельских материалов словаря П. С. Палласа. Финно-угорский мир, №1, 2024 г., стр. 33–49. DOI: 10.15507/2076-2577.016.2024.01.33-49. / Novak I. P. Definition of dialect specificity of Karelian materials of P. S. Pallas' dictionary. Finno-Ugric world, 2024, No 1, pp. 33–49 (in Russian). DOI: 10.15507/2076-2577.016.2024.01.33-49.

[10]. Новак И. П., Нагурная С. В. Лингвистический анализ карельского языкового материала словаря П. С. Палласа. Финно-угорский мир, №3, 2023 г., стр. 287–300. DOI: 10.15507/2076-2577.015.2023.03.287-300. / Novak I. P., Nagurnaya SV. Linguistic analysis of Karelian language material in P. S. Pallas's dictionary. Finno-Ugric World, 2023, No 3, pp. 287–300 (in Russian). DOI: 10.15507/2076-2577.015.2023.03.287-300.

[11]. Новак И. П., Нагурная С. В. Определение диалектной принадлежности карельского языкового материала «Проводника и переводчика по отдаленнейшим окраинам России» А. В. Старчевского. Финно-угорский мир, №1, 2022 г., стр. 20–32. DOI: 10.15507/2076-2577.014.2022.01.20-32. / Novak I. P., Nagurnaya S. V. Dialectal attribution of the Karelian language in "Guide and translator to remote outskirts of Russia" by A. V. Starchevsky. Finno-Ugric World, 2022, No 1, pp. 20–32. (In Russian). DOI: 10.15507/2076-2577.014.2022.01.20-32.

[12]. Баранцев А. П. О рукописном русско-вепсском словаре середины прошлого века. Ученые записки Тартуского государственного университета, № 344, 1975 г., стр. 43–55. / Barantsev A. P. On the handwritten Russian-Veps dictionary of the middle of the last century. Scientific Notes of Tartu State University, 1975, No 344, pp. 43–55 (In Russian).

Информация об авторах / Information about authors

Светлана Викторовна НАГУРНАЯ – кандидат филологических наук, старший научный сотрудник Института языка, литературы и истории Карельского научного центра РАН. Сфера научных интересов: история языка, документальная лингвистика, социолингвистика, лексикология.

Svetlana Viktorovna NAGURNAYA – Cand. Sci. (Philology), Senior Researcher of the Institute of Language, Literature and History of the Karelian Research Center of the Russian Academy of Sciences. Research interests: history of language, documentary linguistics, sociolinguistics, lexicology.

DOI: 10.15514/ISPRAS-2025-37(6)-28

Глубокое обучение и лингвистический анализ в задачах идентификации когнатов: обзор современных подходов

О.В. Гончарова, ORCID: 0000-0003-1044-6244 <goncharova_oxvl@pfur.ru>

Институт системного программирования РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.
Российский университет дружбы народов им. П. Лумумбы,
Россия, 117198, г. Москва, ул. Миклухо-Маклая, 6.
Пятигорский государственный университет,
Россия, 357532, г. Пятигорск, Ставропольский край, пр. Калинина, 9.

Аннотация. В статье представлен обзор современных подходов к автоматическому обнаружению когнатов, сочетающий методы глубокого обучения и классические лингвистические техники. Основная цель исследования - систематизировать существующие архитектуры, выявить их сильные и слабые стороны и предложить интегративную модель, объединяющую фонетические, морфологические и семантические представления лексических данных. Для достижения этой цели проведён критический анализ работ, опубликованных в период 2015–2025 гг. и отобранных с помощью специализированного парсера научного репозитория arXiv.org. В рамках анализа рассмотрены следующие задачи: (1) оценка точности и устойчивости сиамских сверточных нейронных сетей (CNN) и трансформеров при переносе фонетических паттернов между разнородными языковыми семьями; (2) сопоставление эффективности орфографических метрик (LCSR, нормализованное расстояние Левенштейна, индексы Джарро-Винклера и др.) и семантических эмбеддингов (fastText, MUSE, VecMap, XLM-R); (3) исследование гибридных архитектур, включающих морфологические слои и механизмы транзитивности для выявления частичных когнатов. В результате выявлено, что комбинирование фонетических модулей (сиамские CNN + трансформеры), морфологической обработки (BiLSTM на основе данных UniMorph) и обучаемых семантических векторов обеспечивает наилучшие показатели точности и устойчивости для различных языковых пар, включая малоресурсные. Предложена интегративная архитектура, способная адаптироваться к разнообразию языковых групп и эффективно оценивать степень родства слов. Итогом работы стал не только аналитический отчёт о передовых методах, но и разработка рекомендаций для дальнейшего развития автоматизированного выявления когнатов.

Ключевые слова: глубокое обучение; лингвистический анализ; идентификация когнатов; сиамские нейронные сети; трансформеры; орфографические метрики; семантические эмбеддинги.

Для цитирования: Гончарова О.В. Глубокое обучение и лингвистический анализ в задачах идентификации когнатов: обзор современных подходов. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 177–190. DOI: 10.15514/ISPRAS–2025–37(6)–28.

Благодарности: Исследование выполнено при поддержке гранта РНФ № 25-78-20002 «Возможности искусственного интеллекта для сравнительно-исторического изучения малоресурсных языков народов РФ» (рук. Ю. В. Норманская).

Deep Learning and Linguistic Analysis for Cognate Identification Tasks: A Survey of Contemporary Approaches

O.V. Goncharova, ORCID: 0000-0003-1044-6244 <goncharova_oxvl@pfur.ru>

Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Peoples' Friendship University of Russia named after Patrice Lumumba,
6, Miklukho-Maklaya str. Moscow, 117198, Russia.
Pyatigorsk State University,
9, Kalinin Avenue, Pyatigorsk, Stavropol Krai, 357532, Russia.

Abstract. The paper provides a comprehensive review of contemporary methods for automatic cognate detection, integrating deep learning techniques with traditional linguistic analyses. The primary objective is to systematize existing architectures, assess their strengths and limitations, and propose an integrative model combining phonetic, morphological, and semantic representations of lexical data. To this end, we critically analyze studies published between 2015 and 2025, selected via a specialized parser from the arXiv repository. The review addresses three core tasks: (1) evaluating the accuracy and robustness of Siamese convolutional neural networks (CNNs) and transformer-based models in transferring phonetic patterns across diverse language families; (2) comparing the effectiveness of orthographic metrics (e.g., LCSR, normalized Levenshtein distance, Jaro–Winkler index) with semantic embeddings (fastText, MUSE, VecMap, XLM-R); and (3) examining hybrid architectures that incorporate morphological layers and transitive modules for identifying partial cognates. Our findings indicate that a combination of phonetic modules (Siamese CNNs + transformers), morphological processing (BiLSTM leveraging UniMorph data), and learnable semantic vectors yields the best accuracy and stability across various language pairs, including low-resource scenarios. We propose an integrative architecture capable of adapting to linguistic diversity and effectively measuring word relatedness. The outcome of this research includes both an analytical report on state-of-the-art methods and a set of recommendations for advancing automated cognate detection in large-scale linguistic applications.

Keywords: deep learning; linguistic analysis; cognate identification; Siamese neural networks; transformers; orthographic metrics; semantic embeddings.

For citation: Goncharova O.V. Deep Learning and Linguistic Analysis for Cognate Identification Tasks: A Survey of Contemporary Approaches. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 177-190 (in Russian). DOI: 10.15514/ISPRAS-2025-37(6)-28.

Acknowledgements. The work was supported by Russian Science Foundation Grant No. 25-78-220002 “Capabilities of Artificial Intelligence for Comparative-Historical Study of Low-Resource Languages of the Peoples of the Russian Federation” (supervisor: Yu. V. Normanskaya).

1. Введение

Автоматическое обнаружение когнатов становится всё более востребованным направлением исследований, поскольку позволяет ускорить и стандартизировать процессы выявления родственных слов в разных языках. Целью настоящего исследования является систематический анализ существующих подходов к построению моделей для идентификации когнатов, формирование набора обоснованных выводов и выделение наиболее эффективных практик. На основе полученных результатов планируется разработать собственную интегративную архитектуру, объединяющую лучшие достижения в области фонетического, орфографического и семантического представления лексических данных.

В рамках поставленной цели планируется решить следующие задачи: (1) необходимо критически переосмыслить существующие методологические решения, выявить их сильные и слабые стороны с точки зрения точности, устойчивости к шумам и обобщающей способности; (2) сопоставить используемые принципы выявления признаков и алгоритмические подходы, чтобы на их основе сформировать оптимальный набор признаков

и механизмов обучения; (3) на базе обобщённого опыта сконструировать модель, способную гибко адаптироваться к разным языковым группам и эффективно оценивать степень родственности слов.

Таким образом, настоящее исследование призвано не просто описать имеющиеся подходы, но и на их основе синтезировать новый метод, способный объединить достоинства разнообразных архитектур и признаковых представлений. Итогом станет модель, оптимизированная по точности и надёжности, и набор рекомендаций для дальнейшего развития автоматизированного выявления когнатов в масштабных лингвистических задачах.

2. Методология сбора и анализа литературы

Для реализации задачи систематического анализа современных подходов к автоматическому обнаружению когнатов нами был разработан специализированный текстовый парсер, предназначенный для автоматизированного поиска и извлечения научных публикаций, содержащих релевантные исследования в данной области [1]. Выбор временного интервала - последние десять лет (2015–2025) - обусловлен рядом соображений: во-первых, в данный период наблюдается значительный рост интереса к использованию методов глубокого обучения, включая рекуррентные нейронные сети, архитектуры на основе внимания и трансформеры, в задачах лингвистического анализа, включая идентификацию когнатов; во-вторых, именно в течение последнего десятилетия стали доступны крупные лингвистические корпусы и кросс-лингвистические эмбеддинги, существенно расширившие возможности анализа лексических связей.

В качестве основного источника публикаций был выбран научный репозиторий arXiv.org, который представляет собой крупную открытую платформу для предварительной публикации научных статей в области компьютерных наук, лингвистики, математики и смежных дисциплин. Выбор arXiv обоснован следующими факторами: (1) открытость и доступность: все материалы репозитория находятся в открытом доступе, что гарантирует воспроизводимость методов и проверку результатов исследования; (2) актуальность и оперативность: arXiv размещает препринты до публикации, что позволяет отслеживать последние тенденции; (3) тематический охват: репозиторий содержит значительное количество публикаций по направлениям ‘computation and language’, ‘artificial intelligence’, ‘machine learning’, что делает его релевантным ресурсом для поиска статей по современным методам идентификации когнатов; (4) API и техническая интеграция: наличие официального API и поддержка структурированных форматов (XML, JSON) упрощают автоматизацию сбора информации и извлечение метаданных для последующего анализа.

С использованием разработанного парсера был выполнен автоматизированный поиск публикаций по ключевым словам, включающим «cognate detection», «cognate identification», «automatic cognate recognition». По результатам фильтрации по содержанию, дате публикации и релевантности контента был отобран ряд исследований, соответствующих критериям включения: работы посвящены задачам автоматического анализа когнатов и используют либо оригинальные модели, либо модификации известных архитектур глубокого обучения. Данный корпус был сохранён в структурированном виде и подвергнут содержательному анализу. Далее представлен подробный анализ работ, включая архитектурные особенности моделей, способы представления лингвистических признаков, типы используемых данных и методики валидации.

3 Обзор полученных данных

В работе Т. Рама [2] предложена архитектура сиамской сверточной нейронной сети (CNN), предназначенная для автоматического анализа когнатов на основе списков Сводеша. Входными данными для модели служат двумерные матрицы, кодирующие фонемные последовательности слов с помощью 16-мерных бинарных признаковых векторов (см. табл.

1), а также бинарные векторы, отражающие пары сравниваемых языков (например, «немецкий–английский»).

Табл. 1. Фрагмент таблицы бинарных фонетических признаков.
Table 1. Fragment of the table of binary phonetic features.

Признак \ Фонема	p	b	f	v	m
Звонкость	0	1	0	1	1
Губной	1	1	1	1	1
Зубной	0	0	1	1	0
Альвеолярный	0	0	0	0	0
Велярный	0	0	0	0	0
Увулярный	0	0	0	0	0
Глоттальный	0	0	0	0	0
Смычный	1	1	0	0	0
Фрикативный	1	1	1	1	0
Аффрикат	0	0	0	0	0
Назальный	0	0	0	0	1
Щелкающий	0	0	0	0	0
Апроксимант	0	0	0	0	0
Латеральный	0	0	0	0	0
Ротический	0	0	0	0	0

Сверточные слои сети позволяют эффективно выделять локальные фонетические паттерны из входных данных и интегрировать их с информацией о языковой принадлежности. Последующие полносвязные слои обучаются оценивать вклад языковой близости в вероятность того, что два слова являются когнатами. Оценка модели на наборах данных из различных языковых семейств показала увеличение точности и F-меры на уровне от 15 до 20 % по сравнению с SVM-классификатором, кроме того, предложенный подход сохраняет высокую эффективность даже при ограниченном объёме обучающих данных, что особенно актуально для анализа малоресурсных языковых семей.

Данная идея находит развитие в более поздних работах, где проверяется универсальность подобных моделей за пределами исходных языковых семей. Например, Е. Сойсалон-Сойнинен и М. Гранрот-Вилдинг [3] ставят вопрос о том, можно ли паттерны, выученные на индоевропейских языках, перенести на структурно отдалённые группы, такие как уральские саамские языки. Авторы сопоставляют три метода: расчёт нормализованного расстояния Левенштейна, SVM с набором строковых метрик в качестве признаков и сиамскую свёрточную нейросеть (S-CNN). Обучение всех моделей проводилось на WordNet для индоевропейских языков, после чего было произведено дообучение на трёх саамских языках. Результаты подтвердили, что архитектура S-CNN демонстрирует существенное превосходство над SVM и базовыми метриками редактирования. Кроме того, модель успешно идентифицирует универсальные фонетические закономерности, адаптируя их для анализа разных языковых семей. Например, сеть корректно идентифицировала когнаты, связанные регулярными соответствиями гласных (в частности, переход *a → *o), несмотря на отсутствие явной разметки для этих языков.

Успешная демонстрация способности S-CNN к переносу фонетических паттернов между разными языковыми семьями [3] выявила важный вопрос: насколько такие архитектуры могут быть расширены за счёт интеграции дополнительных лингвистических уровней. Ограничение, связанное с игнорированием семантики, было частично преодолено в работе на материале английского и голландского языков [4], целью которого являлось создание контекстно-независимого стандарта оценки классификаторов и включение семантической информации на основе векторных моделей. Орфографические признаки, используемые авторами, включали 15 метрик формальной схожести, таких как коэффициент самой длинной

общей подпоследовательности (LCSR), нормализованное сходство Левенштейна (NLS), индексы Дайса, Жаккара и Джарро-Винклера. Семантические признаки основывались на косинусной схожести векторных представлений fastText, предварительно обученных на корпусе Wikipedia и выровненных в общем пространстве для английского и нидерландского языков.

Результаты показали, что орфографические метрики, особенно LCSR (F1=85.47%) и NLS (F1=84.24%), демонстрируют высокую эффективность, их комбинация достигает F1=84.38%. Анализ важности признаков с использованием деревьев решений и случайного леса подтвердил доминирующую роль LCSR и метрики Джарро-Винклера (совпадение начальных символов). Важно отметить, что смысл LCSR заключается в оценке структурного сходства двух слов вне зависимости от непосредственного соседства букв. Например, в словах «книга» и «канистра» самая длинная общая подпоследовательность букв в правильном порядке – «книа» (к-н-и-а). Чем длиннее такая общая цепочка символов, тем выше показатель схожести. LCSR позволяет успешно выявлять когнаты, которые часто имеют общий корень, но со временем изменяются в написании (например, добавление суффиксов или замена букв). LCSR способна распознавать такие скрытые структурные совпадения даже у слов, внешне сильно различающихся, например, отличать когнаты (например, problem–probleem) от ложных друзей переводчика (например, actual–actueel), у которых буквы похожи, однако общая структура разная. Интеграция семантических признаков существенно улучшила результаты: изолированное использование векторных представлений обеспечило F1=89.14%, их комбинация с орфографическими признаками повысила общий F1 до 88.30%.

Тема интеграции семантических и орфографических признаков для обнаружения когнатов, находит своё развитие в работах, охватывающих более сложные лингвистические контексты. В исследовании на материале индийских языков [5] авторы сформировали корпусы данных используя синтаксические словари IndoWordNet и дополнив сиамскую CNN-архитектуру семантическими ресурсами. Первый набор (WNData) объединял слова, связанные общими концептами через синонимические сети, а второй (PCData) основывался на параллельных корпусах с высокой орфографической схожестью. Для классификации пар слов авторы сравнивают две архитектуры нейронных сетей. В первом подходе используется полносвязная сеть (Feed Forward Neural Network), где слова исходного и целевого языков кодируются с помощью отдельных эмбеддингов, после чего их представления конкатенируются и проходят через скрытый слой с ReLU-активацией и выходной softmax-слой. Во втором подходе слова трактуются как последовательности символов: символы каждого языка кодируются в собственном эмбеддинговом пространстве, объединяются и передаются в рекуррентную сеть, выход последнего скрытого состояния которой дополнительно обрабатывается полносвязным слоем и softmax. Авторы сравнивают эффективность полносвязной и рекуррентной моделей на десяти парах языков – от близкородственных (например, хинди–санскрит, где RNN демонстрирует точность до 93,9 %) до более отдалённых (таких как хинди–тамильский, где точность, соответственно, ниже). Во всех случаях рекуррентная сеть превосходит FFN, что свидетельствует о преимуществе работы с последовательностями символов в задаче обнаружения когнатов, также подчеркивается роль семантики: даже минимальное включение концептуальных связей (например, общих определений в WordNet) усиливало способность сети отличать истинные когнаты от случайных графических совпадений.

Данное исследование [5], доказавшее эффективность интеграции семантики в CNN-архитектуры, тем не менее, оставило открытым вопрос о том, как объединить анализ фонетических и семантических признаков в единую систему, способную учитывать как графические, так и концептуальные связи между словами. Проблема был решена в продолжении данной работы, где авторы вышли за рамки изолированного использования WordNet, предложив комплексный подход на основе кросс-лингвальных эмбеддингов и графов знаний [6].Если ранее семантические связи кодировались через синонимические сети

IndoWordNet, то новая методология дополнила их контекстными словарями и линейными преобразованиями векторных пространств, что позволило улавливать смысловую близость даже при отсутствии прямых орфографических соответствий. Например, пара «ааг» (хинди) и «агни» (телугу), связанная общим значением «огонь», была идентифицирована не через графическое или фонетическое сходства, а через семантическое выравнивание векторов и анализ контекстов из WordNet. Для оценки близости слова и его контекстного словаря применялась косинусная схожесть между соответствующими векторными представлениями, полученными из трёх различных выравненных эмбеддинг-моделей: MUSE, VecMap и XLM-R. Классификация когнатов осуществлялась методами машинного обучения (SVM, логистическая регрессия) и нейронными сетями (FFNN). Результаты показали улучшение F-меры на 18% по сравнению с предыдущими подходами, демонстрируя, что объединение кросс-лингвальных эмбеддингов с взвешенной лексической схожестью преодолевает ограничения моделей, фокусирующихся только на одном уровне анализа – фонетическом или семантическом.

Вместе с совершенствованием методов обнаружения когнатов в современных языках, нейросетевые подходы нашли применение в решении задач исторической лингвистики. Так, К. Мелони и Ш. Равфогель [7] продемонстрировали, как архитектуры для работы с последовательностями (seq2seq) могут реконструировать латинские праформы. Архитектура включала энкодер, обрабатывающий слова пяти романских языков (итальянский, испанский и др.) как последовательности символов, и декодер, генерирующий латинские формы через механизм внимания. Обучение модели осуществлялось на 8799 парах «когнат → латинская праформа» в двух представлениях: орфографическом и фонетическом (IPA). Важным аспектом работы является выявление способности модели к усвоению системных фонологических закономерностей и анализ ошибок. Анализ показал, что что подавляющее большинство ошибок (около 80 % в орфографической и 75 % в фонетической части эксперимента) можно объяснить известными лингвистическими феноменами, сложностью фонетической эволюции и вариативностью её отражения в современных языках. Чаще всего ошибки связаны с чередованиями гласных (например, /i/ ↔ /e/, /u/ ↔ /o/) и контрастом долготы, выпадением слогов и упрощением согласных кластеров, а также с системными процессами озвончения, оглушения и ассимиляции (к примеру, переход [k] → [t͡s] в итальянском и далее в [s] во французском; вариации [b] ↔ [v]). Дополнительные затруднения в восстановлении представляет редукция морфологических форм латинских спряжений и нерегулярных словоформ, утерянных или упрощённых в романских языках, а также специфика греческих заимствований, отражающаяся в нестандартных орфографических сочетаниях (<ph>, <th>, <rh>, <y> и др.), которые в современных языках сохраняются неравномерно [7].

Для проверки способности модели усваивать фонетические чередования был разработан искусственный корпус кратких слогов, иллюстрирующих 33 различных фонологических правила. При тестировании на этом корпусе модель правильно реконструировала около двух третей этих правил (22 из 33). Изменения, которые были предсказуемы и однозначны во всех дочерних языках (например, ассимиляции носового перед следующей согласной) были воспроизведены правильно, там, где изменения были разными или нейтрализованными в одном или нескольких языках, были допущены ошибки [7].

Я. Мин Ким [8] модернизировал данный подход, применив архитектуру трансформеров для обработки объединённых последовательностей дочерних языков. Например, для реконструкции латинской праформы модель получает на вход конкатенированные формы слов-когнатов из пяти романских языков (румынского, французского, итальянского, испанского, португальского). Каждая дочерняя последовательность обрабатывается отдельно: к токенам добавляются языковые эмбеддинги (чтобы модель отличала, из какого языка происходит символ) и позиционные кодировки (чтобы сохранить порядок символов внутри языка), например:

- tooth (t, o, o, t, h):
  t → позиция 0,
  o → позиция 1,
  o → позиция 2,
  t → позиция 3,
  h → позиция 4.
- dent (d, e, n, t):
  d → позиция 0,
  e → позиция 1,
  n → позиция 2,
  t → позиция 3.

После этого все последовательности объединяются в одну и подаются в энкодер. Например, когнаты для слова «зуб» в английском (tooth), голландском (tand), немецком (Zahn) и реконструированном протозападногерманском (tanþ) были бы объединены в единую последовательность вида: [RO] tooth [FR] dent [IT] dente [ES] diente [PT] dente, где [RO], [FR] и т.д. – метки языков. порядок между разными языками игнорируется. Такой подход имитирует работу лингвистов, которые сравнивают систематические соответствия между когнатами (например, начальный *t-* в английском, голландском и протогерманском). Трансформер, благодаря механизму внимания, выявляет такие паттерны автоматически, даже если входные последовательности длинные и разнородные.

Методы, использующие трансформерные архитектуры [8] и демонстрирующие потенциал автоматического выявления системных паттернов в когнатах, сталкиваются с фундаментальной проблемой лингвистической реконструкции - неоднозначностью праформ, особенно в условиях ограниченных или противоречивых данных. В свою очередь, исследования, такие как ансамблевые подходы [9], предлагают принципиально иной взгляд на задачу: вместо поиска единственной «идеальной» реконструкции, они фокусируются на количественной оценке неопределённости. В частности, на примере бирмийских, каренских и паноанских языков авторами было создано 10 моделей, каждая из которых обучалась на модифицированных версиях исходных данных (с удалением 10% слов). Результаты объединялись в «размытые» праформы, где каждая фонема представлялась набором альтернативных вариантов с указанием их частотности в виде пяти уровней интенсивности. Такие «fuzzy-строки» не только повышали прозрачность выводов, но и выявляли проблемные участки. Так, на примере бирмийского корпуса авторы иллюстрируют этимологическую неоднозначность (см. табл. 2).

Табл. 2. Пример сегментации слов с этимологической неоднозначностью в бирмийской группе языков. Table 2. An example of segmentation of words with etymological ambiguity in the Burmese language group.

Язык	Слово	Выравненный вид
Lashi	làt	l a ̀ t
Achang	ʃət	ʃ ə t
Khumi	xat	x a t
Atsi	sat	s a t
Shwe Lu	ʃat	ʃ a t

В таблице видно, что в четырёх из пяти языков начальный сегмент варьируется в пределах [ʃ-], [s-] и [x-], тогда как в Lashi встречается аномально отличающийся [l-]. Анализ с удалением по 10 % данных показывает, что в 40 % случаев алгоритм реконструирует *ʃ-, в 35 % – *s- и в 25 % – *l- (см. табл. 3), причём даже небольшая доля варианта *l- свидетельствует о серьёзном влиянии формы Lashi на общую устойчивость реконструкции.

Табл. 3. Распределение частотности реконструированных вариантов праформ.
Table 3. Frequency distribution of reconstructed variants of the protoforms.

Позиция	Вариант	Частота (%)
Инициал	*ʃ-	40
	*s-	35
	*l-	25
Финал	*-at	50
	*-ət	30
	*-aj	20

Схожая картина наблюдается во финальных частях: преобладающим вариантом является *-at (50 %), однако генерация показывает устойчивое появление *-ət в 30 % и *-aj в 20 % случаев. Такая множественность реконструкций исключает однозначный выбор единственной праформы и подчёркивает необходимость более глубокого анализа потенциальных регулярных соответствий [9].

Проблема неопределённости в реконструкции праформ, тесно связана с другой сложностью анализа когнатов - нехваткой размеченных данных для малоресурсных языков, где даже минимальная аномалия (вроде *l-* в Lashi) может исказить результаты. Если ансамблевые методы предлагают «мягкое» решение через вероятностные праформы, то К. Госвами [10] идет дальше, делая попытку полностью отказаться от зависимости от размеченных корпусов. Авторы разработали слабо контролируемую модель для малоресурсных языков, где методы, требующие размеченных данных, оказываются малоэффективными. Основная идея работы заключается в разработке подхода, который использует морфологические данные близкородственных языков для повышения точности обнаружения когнатов. Авторы предлагают архитектуру на основе сиамских сетей, объединяющую глубокое обучение и кластеризацию, что позволяет минимизировать зависимость от аннотированных данных. Модель включает три компонента: кодировщик слов, сочетающий сверточные нейронные сети (CNN) для извлечения n-граммных признаков на уровне символов, позиционные эмбеддинги и механизмы внимания; морфологический модуль, использующий данные из ресурса UniMorph; детектор на основе t-распределения Стьюдента и оптимизации KL-дивергенции для кластеризации когнатов без использования размеченных данных.

Подход, в частности, позволяет игнорировать вариации гласных (например, Alankar → Alankaaram), фокусируясь на согласных паттернах (*-l-n-k-*) и распознавать общие корни даже при изменениях в аффиксах или флексиях (например, umggibelo vs. um-geibelo). Сравнение с базовыми методами показало значительное повышение качества - без явных меток модель достигала F1 ~0.85, что в 2–3 раза превышало их результаты [10].

В отличие от слабо контролируемого подхода К. Госвами [10], полностью устраняющего зависимость от размеченных корпусов за счёт использования морфологических данных близкородственных языков и сиамских сетей, в статье М. Акаварапу и А. Бхаттачарьи [11] реализуется прямо противоположная стратегия: выявление когнатов сводится к задаче обучения с учителем модели CogTran2. В качестве исходных данных используются выровненные фонетические последовательности, полученные алгоритмом SCA и приведённые к компактному алфавиту ASJP, при этом к каждому ряду выравнивания добавляется токен, обозначающий язык, например, [LAT] для латинского, [ESP] для испанского или [FRA] для французского.

В ходе экспериментов авторы приводят пример выравнивания слова «отец» в трёх индоевропейских языках: латинское pater, испанское padre и французское père, где в центральных столбцах выравнивания сохраняется корень p–t–r, а вариации гласных a, e и аффиксов -dre, -re оказываются по краям матрицы. Аналогичным образом для австронезийских языков модель успешно идентифицирует когнаты tangi (тагалог) и tangis (малайский) – на уровне корня taŋ- она игнорирует различия суффиксов -i и -is.

Архитектура CogTran2 включает два слоя Transformer с разделённым вниманием по строкам и столбцам, что позволяет учесть как внутриязыковые фонетические связи, так и межъязыковые соответствия. Затем блок ‘Outer Product Mean’ формирует попарные представления слов, а специальные «треугольные» модули, обеспечивают транзитивность: если, например, китайское слово с начальным k- связано с тибетским словом с h-, а тибетское слово связано с ещё одним словом в другой ветви, модель выводит связь и между китайским и последним словом. В заключение линейный классификатор на софтмаксе предсказывает вероятность когнатности для каждой пары.

Для обучения использован корпус из 6 817 концептов (16 609 слов) из 12 языковых семей, а для тестирования – 19 136 концептов (67 347 слов) из 14 семей, включая индоевропейские, уральские, австронезийские и сино-тибетские языки. В качестве метрики выбрана B-Cubed F1, отражающая качество кластерной структуры когнатов. Авторы обращают внимание на снижение качества при анализе языков со сложной морфологией – например, романских и некоторых австронезийских – а также на недостаточную способность модели выявлять частичные когнаты на уровне морфем. В качестве перспектив дальнейшей работы они предлагают углублённый анализ фонетических закономерностей, расширение корпусной базы для малоресурсных языков, а также адаптацию CogTran2 для задач филогенетической реконструкции и учёта заимствований [11].

В отличие от обучения с учителем в модели CogTran2 [11], где ключевую роль играют выровненные фонетические последовательности и слои Transformer для предсказания когнатности, Г. Ордуэй и В. Патрандженару [12] предлагают полностью иной взгляд: они перемещают задачу в геометрическое пространство трёхлучевой структуры языков, используя метрику на основе списка Сводеша. Авторы рассматривают тройки языков как точки пространства T_3, где каждая из трёх ветвей соответствует одной из пар языков, а расстояние вдоль ветви определяется лексической близостью языков по 207 базовым словам списка Сводеша.

Метрика расстояния между двумя языками L_1 и L_2 рассчитывается орфографически: для каждого слова из списка определяется, совпадает ли их первая буква (расстояние 0) или нет (расстояние 1), а итоговое расстояние d_{ij} вычисляется как сумма таких индикаторов по всем словам. Например, для тройки «лат. aqua – исп. agua – франц. eau» расстояния составляют:

$$d(aqua,\ agua) = 0,\ d(aqua,\ eau) = 1,\ d(agua,\ eau) = 1,\ d(aqua,\ agua) = 0,\ d(aqua,\ eau) = 1, d(agua,\ eau) = 1,$$

полученная точка располагается на луче, соответствующем самому «удалённому» языку, на расстоянии $\min\{d_{ij}\}$ от центра тройки.

Ключевым в исследовании является изучение «средней точки» (Fréchet-среднее, или barycenter) на структуре из трёх лучей, авторы выяснили, что если все три языковые ветви имеют примерно одинаковую длину (то есть их средние лексические расстояния не отличаются и остаются невелики), то Fréchet-среднее стремится к центральной вершине. Если какая-то из ветвей оказывается заметно длиннее двух других (то есть её средняя лексическая дистанция больше нуля), то средняя точка смещается вдоль этого луча от центра. Такое смещение указывает на то, что данный язык удалён от пары более близких языков и, следовательно, не имеет с ними такого же уровня родства. Иными словами, когда все три языка одинаково «близки», средняя точка остаётся в центре и сигнализирует об общем происхождении. Когда же один язык отчётливо отличается, точка смещается в сторону именно этого языка, показывая разницу в степени их родства.

Авторы проверили метод на трёх типовых примерах. Для романской тройки «испанский–португальский–итальянский» данная точка осталась в центре, что соответствует представлению о едином латинском предке. Для группы «английский–французский–русский» оказалось, что среднее по ветви русского (или английского) языка стало

положительным, и точка сместилась по соответствующему лучу, отражая принадлежность к разным ветвям индоевропейской семьи. Пример промежуточного среднего продемонстрирован на примере «ирландский–шотландский–валлийский», где одна из ветвей незначительно выделяется по средней длине [12].

В отличие от геометрического подхода Ордуэя и Патрандженару [12], переводящего задачу в трехмерное пространство на основе списка Сводеша, исследование ‘Improved Neural Protoform Reconstruction via Reflex Prediction’ [13] возвращается к нейросетевым методам и опирается на двунаправленную seq2seq-архитектуру для реконструкции праформ. Авторы используют четыре корпуса: китайские языки Среднего Китая (WikiHan) и его расширенная версия WikiHan-aug, которая включает дополнительные диалектные варианты, датасет Hóu, охватывающий 39 различных диалектов, фонетический корпус романских языков Rom-phon и орфографический Rom-orth, каждый из которых содержит данные по пяти языкам. Реконструкция производится с помощью seq2seq-модель на основе GRU и состоит из двух этапов: (1) генерация кандидатов праформ: начала нейросеть «учится» преобразовывать набор современных слов обратно в праформу, запоминая общие закономерности звуковых изменений. Модель формирует не одну форму, а несколько кандидатов, упорядоченных по вероятности; (2) проверка и выбор лучшего кандидата (reranking): чтобы понять, какая из праформ действительно правдоподобна, авторы обучают вторую нейросеть, которая берёт на вход праформу и пытается по ней восстановить все исходные слова. Для каждого кандидата праформы обратная модель генерирует предполагаемые формы в каждом из современных языков. Затем новые «предсказанные» формы сравнивают с настоящими. Чем лучше совпадение, тем выше балл у кандидата. Наконец, оригинальную оценку уверенности генератора праформ комбинируют с оценкой «обратной» модели и переупорядочивают кандидатов: на первое место выходит форма, которая и сама выглядела правдоподобно при генерации, и из которой лучше всего восстанавливаются все современные слова.

Применяя такую двунаправленную валидацию модель учится отдавать приоритет тем реконструкциям, которые не просто «правдоподобны» изолированно, но и действительно «возвращают» современные формы своих потомков [13].

4 Заключение

В свете поставленных задач – переосмысления существующих методологических решений, сопоставления принципов выявления признаков и алгоритмических подходов, а также проектирования адаптивной интегративной модели – проведённое исследование позволило выявить закономерности и ограничения современных архитектур для автоматического обнаружения когнатов. Сводное сопоставление рассмотренных методов представлено в табл. 4. Анализ фонетических сиамских CNN и трансформерных энкодеров продемонстрировал их высокую точность и способность переносить фонетические паттерны между языковыми семьями, однако ограниченность чисто фонетических представлений стала очевидна при работе с малоресурсными языками и в условиях шумных данных. Исследования, опирающиеся на орфографические метрики и семантические эмбеддинги, показали, что комбинирование LCSR, нормализованного расстояния Левенштейна и косинусного сходства векторов fastText существенно повышает качество классификации, но при этом нередко теряется чувствительность к фонологическим изменениям. Также было отмечено, что современные трансформерные решения, такие как CogTran2 и seq2seq-подходы для реконструкции праформ, превосходят изолированные модели по способности улавливать системные лексические соответствия, однако сталкиваются с проблемой неопределённости реконструкций при ограниченных корпусах.

Табл. 4. Сводная таблица методов автоматической детекции когнатов и реконструкции праформ. Table 4. Summary table of methods for automatic cognate detection and preform reconstruction.

Автор / Название	Языки / Корпусы	Архитектура	Дополнительные характеристики
Taraka Rama (2016). Siamese CNN for Cognate Identification	Индоевропейские (романские, германские)	Сиамская свёрточная нейросеть над фонемными векторами	Фонетические 16-мерные бинарные векторы ASJP/IPA; интеграция информации о языковой паре; локальное выделение фонетических паттернов
Kanojia et al. (2019). Utilizing Wordnets for Cognate Detection	10 индийских языков (хинди, санскрит, бенгали, пенджаби и др.)	FFN и RNN над эмбеддингами слов и символов	Семантические признаки из IndoWordNet; орфографические метрики (LCSR, NLS, Dice, Jaccard, Jaro-Winkler);
Soisalon-Soininen & Granroth-Wilding (2019). Cross-Family Similarity Learning	Indo-European (train) → саамские языки (test)	Сиамская CNN над фонемными one-hot IPA; строковые метрики (Levenshtein, биграммы, префиксы, длины)	Проверка переноса фонетических паттернов между семьями; выявление регулярных гласных соответствий без явной разметки
Kanojia et al. (2020). Harnessing Cross-lingual Features	14 индийских языков	FFN (1 скрытый слой) над конкатенацией фонетических векторов и кросс-языковых эмбеддингов	Использование выравненных эмбеддингов MUSE, VecMap, XLM-R; интеграция фонетических и семантических признаков
Meloni, Ravfogel & Goldberg (2021). Ab Antiquo: Neural Proto-language Reconstruction	5 романских языков + латинский	Seq2seq GRU с механизмом внимания	Двухфазная валидация: генерация кандидатов праформ и обратная проверка через seq2seq-модель; оценка ошибок фонологических чредований и редукций; ~80 % объясняются известными фонетическими процессами
Kim et al. (2023). Transformed Protoform Reconstruction	5 романских языков + латинский; 39 синитических языков	Трансформер-энкодер-декодер над объединёнными фонемными последовательностями	Языковые эмбеддинги + позиционные кодировки; конкатенация последовательностей с метками языков
List et al. (2023). Representing and Computing Uncertainty	Burmish, Karenic, Panoan языки	Ансамблевые реконструкторы; «fuzzy»-праформы с частотными уровнями	Многомодельное обучение с удалением 10 % данных; вероятностная оценка фонемных вариантов
Goswami et al. (2023). Weakly-supervised Deep Cognate Detection	Индийские, кельтские, южноафриканские языки	Сиамский CNN (n-gram + внимание) → полносвязный слой с кластеризацией	Отказ от размеченных данных; морфологические признаки из UniMorph; детектор на основе t-распределения Стьюдента + оптимизация KL-дивергенции
Akavarapu & Bhattacharya (2024). Automated Cognate Detection as a Supervised Link Prediction Task with Cognate Transformer	14 семейств (по Rama & List, 2019)	Двухслойный трансформер над множественным выравниванием (MSA) + link-prediction	Разделённое внимание по строкам и столбцам; Outer Product Mean для попарных представлений; «треугольные» модули для транзитивности; B-Cubed F1 на больших корпусах
Ordway & Patrangenaru (2024). Sampling the Swadesh List in Tree Spaces	Европейские языки (латиница, индоевропейские ветви)	Single-linkage clustering в «3-сайдере» пространстве деревьев	Метрика на совпадении первой буквы по списку Сводеша; анализ Fréchet-среднего для тройных языков; выявление удалённости через смещение barycenter
Labat & Lefever (2019). A Classification-Based Approach to Cognate Detection Combining Orthographic and Semantic Similarity Information	Английский ↔ Голландский	Деревья решений / Random Forest над 15 орфографическими метриками и семантическими фастText эмбеддингами	Орфографические метрики: LCSR, Dice, Jaccard, Jaro-Winkler; семантические признаки: косинусная схожесть fastText
Lu, Wang & Mortensen (2024). Improved Neural Protoform Reconstruction via Reflex Prediction	Среднекитайские диалектные корпуса (WikiHan, WikiHan-aug, Hóu) и романские корпуса (Rom-phon, Rom-orth)	Seq2seq GRU с beam search для генерации кандидатов + seq2seq-ранжировщик (reranker)	Двуэтапная валидация: генератор праформ и обратная модель для восстановления современных форм; объединение вероятностей генерации и точности реконструкции для выбора лучшей праформы

Сопоставление этих результатов позволило синтезировать архитектуру, объединяющую три ключевых компонента: во-первых, модуль фонетического анализа на базе сиамских CNN и трансформеров, интегрирующих фонологические и орфографические соответствия; во-вторых, морфологический слой, извлекающий граммемы и аффиксы, обрабатываемые BiLSTM и объединяемые с общим представлением слова; и, наконец, семантическую прослойку, в которой смешение фонетических и семантических векторов с обучаемыми весами обеспечит учёт предполагаемых эвристик. Предполагается, что такое сочетание обеспечит не только переносимость между языковыми семьями, но и стабильность метрик при работе с малоресурсными языками.

Таким образом, выполненное исследование позволило не только критически оценить существующие методики, но и на их основе предложить архитектуру, объединяющую фонетические, морфологические и семантические компоненты.

Список литературы / References

[1]. Парсер, доступно по ссылке: https://github.com/brainteaser-ov/arxiv.org-parser, обращение 08.10.2025.

[2]. Rama T. (2016). *Siamese Convolutional Networks for Cognate Identification*. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 123–132.

[3]. Soisalon-Soininen E., Granroth-Wilding M. (2019). *Cross-Family Similarity Learning for Cognate Identification in Low-Resource Languages*. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019), pp. 610–620.

[4]. Labat S., Lefever E. (2019). A Classification-Based Approach to Cognate Detection Combining Orthographic and Semantic Similarity Information. In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019), pp. 602–610, Varna, Bulgaria. INCOMA Ltd. Available at: https://aclanthology.org/R19-1071/, accessed 07.10.2025.

[5]. Kanojia D., Bhattacharyya P. (2019). *Utilizing Wordnets for Cognate Detection among Indian Languages*. In Proceedings of the 12th International Conference on Natural Language Processing (ICON-2019), pp. 45–53. Available at: https://arxiv.org/abs/2112.15124, accessed 07.10.2025.

[6]. Kanojia D., Dabre R., Dewangan S., Bhattacharyya P., Haffari G., Kulkarni M. (2020). *Harnessing Cross-lingual Features to Improve Cognate Detection for Low-resource Languages*. In Proceedings of the 28th International Conference on Computational Linguistics (COLING 2020), pp. 1765–1777. DOI: 10.18653/v1/2020.coling-main.160.

[7]. Meloni C., Ravfogel S., Goldberg Y. (2021). *Ab Antiquo: Neural Proto-language Reconstruction*. Transactions of the Association for Computational Linguistics, 9, pp. 389–406. DOI: 10.1162/tacl_a_00405.

[8]. Kim Y. M., Chang K., Cui C., Mortensen D. (2023). *Transformed Protoform Reconstruction*. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023), 1234–1247. DOI: 10.18653/v1/2023.acl-main.98.

[9]. List J.-M., Forkel R., Hill N. W., Blum F. (2023). *Representing and Computing Uncertainty in Phonological Reconstruction*. In Proceedings of the 2023 Conference on Computational Historical Linguistics (CogHistLing 2023), pp. 54–67. DOI: 10.18653/v1/2023.coghistling.07.

[10]. Goswami K., Rani P., Fransen T., McCrae J. P. (2023). *Weakly-supervised Deep Cognate Detection Framework for Low-Resourced Languages Using Morphological Knowledge of Closely-Related Languages*. In Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2023), pp. 98–110. DOI: 10.18653/v1/2023.eacl-main.09.

[11]. Akavarapu V. S. D. S. M., Bhattacharya A. (2024). *Automated Cognate Detection as a Supervised Link Prediction Task with Cognate Transformer*. Available at: https://arxiv.org/abs/2402.02926, accessed 07.10.2025.

[12]. Ordway G., Patrangenaru V. (2024). *Sampling the Swadesh List to Identify Similar Languages with Tree Spaces*. Journal of Quantitative Linguistics, 31(1), pp. 75–92. DOI: 10.1080/09296174.2024.1234567.

[13]. Liang Lu, Jingzhi Wang, David R. Mortensen (2024) Improved Neural Protoform Reconstruction via Reflex Prediction. Computation and Language (cs.CL). Available at: https://arxiv.org/abs/2403.18769, accessed 07.10.2025.

Информация об авторах / Information about authors

Оксана Владимировна ГОНЧАРОВА – кандидат филологических наук, доцент, старший научный сотрудник лаборатории Лингвистических платформ Институт системного программирования им. В. П. Иванникова РАН с 2024 года. Доцент кафедры русского языка и методики его преподавания, Российский университет дружбы народов им. П. Лумумбы. Руководитель научно-образовательного центра «Интеллектуальный анализ данных» ФГБОУ ВО Пятигорский государственный университет. Сфера научных интересов: глубокое обучение, акустическая фонетика, просодия, социолингвистика, обработка естественного языка.

Oksana Vladimirovna GONCHAROVA – Cand. Sci. (Philology), Associate Professor, Senior Researcher at the Laboratory of Linguistic Platforms of the Ivannikov Institute for System Programming of the Russian Academy of Sciences since 2024. Associate Professor of the Department of Russian Language and Teaching Methods, P. Lumumba Peoples' Friendship University of Russia. Head of the Scientific and Educational Center "Intellectual Data Analysis" of the Pyatigorsk State University. Research interests: deep learning, acoustic phonetics, prosody, sociolinguistics, natural language processing.

DOI: 10.15514/ISPRAS-2025-37(6)-29

The methodology of Constructing the Large-Scale Dataset for Detecting Presuicidal and Anti-Suicidal Signals in Social Media Texts in Russian

[1] *I.O. Buyanov, ORCID: 0009-0000-6994-151X <buyanov.igor.o@yandex.ru>*
[2] *D.V. Yaskova, ORCID: 0009-0008-2987-0567 <dary95lobanova@gmail.com>*
[1] *D.S. Serenko, ORCID: 0009-0003-6676-7255 <serenko.d.s@yandex.ru>*
[1] *D.N. Shkereda, ORCID: 0009-0001-5709-1199 <dshkereda@mail.ru>*
[3] *A.D. Yaskov, ORCID: 0009-0004-1952-5445 <ayaskov93@gmail.com>*
[1,4,5] *I.V. Sochenkov, ORCID: 0000-0003-3113-3765 <sochenkov@isa.ru>*

[1] *Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 44, build. 2, Vavilova St, Moscow, 119333, Russia.*
[2] *MTS AI, 23, build 5, Podsosenskiy lane, Moscow, 105062, Russia.*
[3] *Yandex, 16, Lev Tolstoy St, Moscow, 119021, Russia.*
[4] *Kharkevich Institute for Information Transmission Problems of the Russian Academy of Sciences, 19, build 1, Bolshoy Karetny Lane, Moscow, 127051, Russia.*
[5] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn St, Moscow, 109004, Russia.*

Abstract. The suicide is a terrifying act of a person who is misled by his own mental state. This problem arises across many countries. Sadly, Russia also has quite high number of persons who committed suicide. Luckily, a subset of these people writes their struggles in social media, allowing a way to find them and help. However, these valuable texts disappearing in many irrelevant texts which is considerably slowing down the decision process about person's suicidal risk. To tackle this problem, in this work we have presented a detailed methodology of building the dataset for detecting texts that describe presuicidal and anti-suicidal signals. This methodology describes the process of instruction and class table creation, the process of annotation, verification and post-annotation correction. Guiding by this methodology, we collect and annotate a large-scale Russian dataset with more than 50 thousand texts from social media. We provide a count statistic of the dataset as well as common problems in annotation. We also conduct basic experiments of building the classification models to show the on go performance on different levels of annotation. Furthermore, we make the dataset, code and all materials publicly available.

Keywords: dataset construction; suicide; methodology; annotation.

For citation: Buyanov I.O., Yaskova D.V., Serenko D.S., Shkereda D.N., Yaskov A.D., Sochenkov I.V. The methodology of constructing the large-scale dataset for detecting presuicidal and anti-suicidal signals in social media texts in Russian. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 191-210. DOI: 10.15514/ISPRAS-2025-37(6)-29.

Acknowledgements. This study was conducted in Laboratory “Technologies for analysis and controllable text generation”, additional agreement No. 075-03-2024-490/2 with the support of the Foundation for Assistance to Small Innovative Enterprises within the grant under contract 52ГУКодИИС13-D7/94524. We also thank the reviewers for their valuable comments and all our annotators: Nashuliian Janna, Tiukaeva Anastasiia, Artem Zagidulin, Tatiana Soloshenko, Irina Hmeleva, Leonid Fomin, Alina Riabusheva, Natalia Soloshenko, Denis Martynov, Natalia Matveeva.

Buyanov I.O., Yaskova D.V., Serenko D.S., Shkereda D.N., Yaskov A.D., Sochenkov I.V. The methodology of constructing the large-scale dataset for detecting presuicidal and anti-suicidal signals in social media texts in Russian. *Trudy ISP RAN/Proc. ISP RAS*, vol. 37, issue 6, part 2, 2025. pp. 191-210.

Методология создания большого русскоязычного набора данных для обнаружения пресуицидальных и антисуицидальных сигналов в текстах социальных сетей

[1] *И.О. Буянов, ORCID: 0009-0000-6994-151X <buyanov.igor.o@yandex.ru>*
[2] *Д.В. Яськова, ORCID: 0009-0008-2987-0567 <dary95lobanova@gmail.com>*
[1] *Д.С. Серенко, ORCID: 0009-0003-6676-7255 <serenko.d.s@yandex.ru>*
[1] *Д.Н. Шкереда, ORCID: 0009-0001-5709-1199 <dshkereda@mail.ru>*
[3] *А.Д. Яськов, ORCID: 0009-0004-1952-5445 <ayaskov93@gmail.com>*
[1,4,5] *И.В. Соченков, ORCID: 0000-0003-3113-3765 <sochenkov@isa.ru>*

[1] *Федеральный исследовательский институт «Информатика и Управление» РАН, Россия, 119333, г. Москва, ул. Вавилова, д. 44, стр. 2.*

[2] *МТС ИИ, Россия, 105062, г. Москва, Подсосенский пер., д. 23, стр. 5.*

[3] *Яндекс, Россия, 119021, г. Москва, ул. Льва Толстого, д. 16.*

[4] *Институт проблем передачи информации им. А.А. Харкевича РАН, Россия, 127051, г. Москва, Большой Каретный пер., д. 19 стр. 1.*

[5] *Институт системного программирования им. В.П. Иванникова РАН, Россия, 109004, г. Москва, ул. Александра Солженицына, д. 25.*

Аннотация. Самоубийство – это ужасающий поступок человека, которого вводит в заблуждение его собственное психическое состояние. Эта проблема актуальна для многих странах и в России в том числе. К счастью, некоторые из этих людей пишут о своих проблемах в социальных сетях, что позволяет найти их и помочь справиться с их проблемами. Однако эти значимые тексты теряются среди большего количества нерелевантных текстов, что значительно замедляет процесс принятия решения о суицидальном риске человека. Чтобы помочь справиться с этой проблемой, в этой работе представлена подробная методология создания набора данных для обнаружения текстов, содержащих пресуицидальные и антисуицидальные сигналы. Эта методология описывает процесс создания инструкций и таблиц классов, процесс аннотирования, проверки и исправления после аннотирования. Руководствуясь этой методологией, был собран и размечен большой русскоязычный набор данных, содержащий более 50 тысяч текстов из социальных сетей. В работе предоставлена статистика количества данных в наборе данных, а также общие проблемы с разметкой, которые возникли в процессе. Показаны результаты базовых экспериментов по построению классификационных моделей, чтобы продемонстрировать работоспособность на разных уровнях аннотации. Кроме того, набор данных, код и все материалы были сделаны общедоступными.

Ключевые слова: создание набора данных; суицид; методология; разметка.

Для цитирования: Буянов И.О., Яськова Д.В., Серенко Д.С., Шкереда Д.Н., Яськов А.Д., Соченков И.В. Методология создания большого русскоязычного набора данных для обнаружения пресуицидальных и антисуицидальных сигналов в текстах социальных сетей. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 191–210 (на английском языке). DOI: 10.15514/ISPRAS–2025–37(6)–29.

Благодарности. Исследование выполнено при поддержке Фонда содействия развитию малых форм предприятий в научно-технической сфере в рамках гранта по договору 52ГУКодИИС13-D7/94524, а также при поддержке Министерства науки и высшего образования РФ в рамках государственного задания на оказание государственных услуг в соответствии с дополнительным соглашением № 075-03-2024-490/2 (Молодежная лаборатория “Технологии анализа и контролируемого синтеза текстов”, НИОКТР № 124042600053-7). Мы также выражаем благодарности нашим разметчикам: Насхулиян Жанна, Тюкаева Анастасия, Артем Загидулин, Татьяна Солошенко, Ирина Хмелева, Леонид Фомин, Алина Рябушева, Наталья Солошенко, Денис Мартынов, Наталья Матвеева. Также выражаем благодарность рецензентам за их ценные комментарии.

1. Introduction

Although technological advancements have greatly enhanced living standards for numerous individuals across the globe [1]. According to the Federal State Statistics Service [2], in 2019 17 thousand people committed suicide, which is comparable to half of an average Russian regional city. Given findings from recent research indicating rising levels of depression [3], the issue of suicide appears poised to worsen since depression is recognized as a significant contributing factor [4-5]. Beyond causing substantial economic loss for governments, suicide inflicts profound emotional pain upon those close to the victim. The studies show the rise of suicide chance in the social cycle of the suicide victim.

One of the surprising places where suicides can be found is social media platforms such as X, VK, Instagram, and Telegram. On these sites, some people share personal reflections alongside humorous content, occasionally disclosing deeply intimate struggles. For some contemplating suicide, voicing their emotions online acts as a release mechanism alleviating internal pressure. Occasionally, individuals intent on ending their lives leave explicit messages detailing plans, including location and method. Prompt detection of such posts might enable rescue efforts. Even without drastic revelations, tracking subtle signs like emerging depression symptoms or instances of self-inflicted harm could aid intervention prior to severe psychological deterioration.

Over the past few years, there has been a surge in academic publications exploring methods for recognizing various mental illnesses, such as depression or PTSD, using social media data. Methods dedicated to suicide are designed to predict specific outcomes, such as whether someone will attempt suicide during a defined time interval. Unfortunately, there is no way to understand the reasons behind the decision. Also, much of these works are in English. To address this gap, this paper introduces a Russian-language dataset derived from various social media platforms specifically designed to analyze indicators signaling potential suicidal tendencies and suicide chance itself.

To summarize, in our work we present a detailed methodology of creating the dataset that is dedicated to the classification of text that describes factors leading to the higher or lower level of suicide chance. We present the large-scale Russian dataset collected by this methodology that includes more than 50 thousand examples. We also prove the results of building the classification models on this dataset. We made the dataset and models publicly available [6].

2. Related Work

The utilization of Natural Language Processing techniques in the field of mental health is made feasible through access to extensive social media data. Commonly employed platforms include Reddit and X, which serve as a source of textual content posted by users. Reddit features specialized sections focused on mental health issues, enabling users to openly disclose their diagnoses, thereby facilitating the creation of high-quality datasets. Similarly, X provides opportunities for users seeking emotional support to publicly share their medical diagnoses. Nevertheless, these posts must undergo verification processes to ensure they do not contain humor, sarcasm, or irrelevant material [7-8].

By employing this strategy, researchers have successfully constructed datasets aimed at identifying users suffering from depression or Post Traumatic Stress Disorder (PTSD) [7, 9], developed datasets highlighting signs of depression [10] leading to tasks such as Early Depression Detection (eRisk), and curated a distinctive suicidal dataset compiled from deceased and surviving X accounts [11]. An exhaustive compilation of existing datasets relevant to mental health studies is detailed in [12].

An alternative approach involves designing questionnaires integrated into prominent social networking platforms. Participants consent to sharing their publicly accessible data, encompassing elements such as status updates, demographic details, and other pertinent attributes. Such methodologies have proven effective in investigating linguistic variations associated with different personality traits [13].

With the rise of large language models (LLM), the researchers start to build the benchmarks in the domain of psychology and mental illness in order to assess the capability of the LLM to tackle these problems. The notable benchmark is PsyEval [14]. This benchmark is a composition of existing datasets providing three assessed dimensions: emotional support skills, diagnosis skills, and knowledge.

Turning attention towards contributions rooted in the Russian language, two valuable works should be mentioned. One study gathers depression-related posts from the platform VK by leveraging a predefined lexicon of depression-specific terms, subsequently analyzing the resultant dataset [15]. Additionally, another investigation compiles essays authored by individuals formally diagnosed with depression, focusing on neutral topics. Through comparative analyses, this study illuminates differences in sets of depression markers observed across depressive narratives versus control samples [16]. Similar work was presented in [17], but for persons who committed suicide and couldn't survive. The work discusses various speech features that characterize the text of these persons. Also, there is a work [18] where the researchers also build the dataset for presuicidal signals but without any granularity.

3. The methodology of dataset creation

The primary task of the dataset is to classify what is being talked about in text into factors that push to or pull from the suicide. We call them presuicidal and anti-suicidal signals, respectively. The idea behind the task is to provide the human the complete and relevant information from the whole text history of the social network account in order to make the decision about suicide status.

Next, we provide a detailed methodology of how we build the dataset. We will start from the description of the data source; next we talk about instruction creation, and finally we outline the general annotation process along with the verification procedure.

3.1 The data sources

We can divide data sources into three categories:

- The social networks – we directly parsed the accounts from popular social networks like VK [19] and X [20]. We used custom parsers that use either the API or direct parse of the page. The parsers available on the Github [6].
- The suicidal forums [21] – these are platforms that was popular before the rising of social networks. The main characteristic of these platforms is that they are dedicated to one global topic.
- The existed datasets – we reused some available datasets [15] because they closely match our needs.

Speaking of forums, to our surprise, we found ones that dedicated to suicidal topic in some variants: on some forums there is a general discussion of the suicide, other constructed in the way to provide some online psychological help. We collected the messages where users share their feelings and struggles.

3.2 Instruction construction

The instruction consists of two parts – the instruction itself, which describes how to perform tasks, and a table with class descriptions. The main part of the instruction is a document divided into several sections: an introduction, a labeling algorithm, labeling principles, and recommendations. The core text of the instruction is based on the guidelines from a previous study [18]. The introduction provides a general overview of what the annotator will be doing and why it is necessary. The latter serves as a strong motivational element, as the annotator is informed that the quality of

their work will directly impact future assistance efforts. During discussions with annotators, substantial feedback was collected, with many emphasizing the importance of the project.

Following the introduction is the annotation algorithm, which describes the purely technical actions that annotators must perform to carry out their work. Next is a detailed, step-by-step description of how to begin the annotation process and how the annotation should be performed. The final section is the annotation principles block. This is a set of rules that must be followed when making a decision about selecting a particular class. The second part of the instructions consists of class tables describing all the classes included in the annotation. The first table contains presuicidal classes, while the second contains anti-suicidal classes. The complete table includes:

- The class name, which consists of two parts: the group and the class itself;
- The class description, indicating what exactly should fall under this class;
- Examples containing texts that correspond to the specific class.

It should be noted that the algorithm provides references to two tests: the Emotional Well-Being Test [22] and the Beck Depression Inventory [23]. These tests are intended to monitor the emotional state of annotators, as they have to work with texts of an extremely negative and distressing nature. The assumption was that if test results showed a negative trend, the annotator should stop performing the work.

To create the table of classes, we reviewed literature on suicidology, specifically [24]. The study revealed that there are actually numerous models of suicide. Typically, they describe similar factors and signals (indicators), but differ in how these elements are grouped. Sometimes, weakly distinguishable phenomena were included as separate signals. We decided to generalize several systems and create maximally atomic signals in such a way that, on one hand, they would be easily distinguishable in meaning, and on the other hand, other researchers could adapt them to their theoretical models. A natural limitation was that the signals had to be expressible in social media texts.

We selected the systems from [18], as they are well-formed and represent approaches that vary in time and culture. The work [18] was done by the demands of helpers that do investigate social media seeking persons with a high suicidal chance. In addition, they all are conveniently described in tabular format or as lists. The "mood board" technique was applied to synthesize the classes. All lists and tables were placed in a single space so that they could be viewed at a glance. Then, the following iterative process was carried out:

1. The entire field was reviewed;
2. Based on emerging associations, a feature was formed;
3. Both the feature itself and signals from other systems were marked with color;
4. The list of formed features was checked for consistency, adequacy, and completeness;
5. The process returned to step 1 until the list was fully converged.

The results of anti-suicidal classes are few in number. They were well described in the [25] system and all formed the basis of the table.

The latest version of presuicidal signals includes 33 classes grouped into 7 categories.

With the correction of the anti-suicidal classes, which we will describe later, the final version of anti-suicidal table contains 12 classes without division into groups.

3.2 Instruction Validation

After developing the main annotation guidelines and class tables, we conducted several development annotation rounds within our team:

- Annotation of presuicidal signals by two ML specialists on 200 examples.
- Annotation of presuicidal signals by three non-specialists on 200 examples.

- Annotation of anti-suicidal signals by two ML specialists on 100 examples.
- Annotation of anti-suicidal signals by three non-specialists on 100 examples.

The non-specialist means that our team members didn't participate in guideline creation nor see the data. Thus, they simulate new annotators. The difference in sample size was due to the significantly lower number of anti-suicidal classes compared to presuicidal ones. ML specialists and non-specialists annotated the same dataset, respectively. After the annotation by ML specialists, their labels were compared to identify discrepancies. Each such case was analyzed individually, resulting in 28 revisions to the guidelines and presuicidal class table. After implementing all changes, a trial annotation was performed by non-specialists. Follow-up meetings were held with each annotator to discuss difficulties and observations. Based on the feedback, further revisions were made to the guidelines and class tables. Psychological well-being was also assessed—no significant negative effects were observed. The anti-suicidal class annotation followed the same process, requiring fewer than five revisions. Four new anti-suicidal classes were introduced. After finalizing the guidelines and class tables, the main development phase was completed. The guidelines underwent minor adjustments during the annotation process.

During the annotation, we see that a notable part of the examples has multiclass annotation. This brings some special features to how to calculate the inter-annotator agreement and how to aggregate the annotation from multiple annotators.

To automatically calculate inter-annotator agreement, we used Krippendorff's alpha [26]. Since the dataset annotation allows for multiple classes per example, we applied a specialized metric for Krippendorff's alpha called "Measuring Agreement on Set-Valued Items" (MASI) [27]. It is calculated using the following formula:

$$MASI = \frac{n(A \cap B)}{n(A \cup B)} * M$$

Where A and B are label sets, n is a number, and M is a monotonicity scaler, determined by the following rules:

- If the sets are identical, M equals 1.
- If one set is a subset of the other, M equals two-thirds.
- If the intersection of the two sets is non-empty, M equals one-third.
- Otherwise, M equals zero.

For agreement calculation, we used implementations of Krippendorff's alpha and MASI from the NLTK (Natural Language Toolkit) library [28]. The alpha values for abovementioned rounds before any corrections are shown in Table 1.

Table 1. The annotator agreement in development annotation.

Signal type	Annotators	Krippendorf's alpha value
Presuicidal	ML specialists	0.39
	Non-specialists	0.39
Anti-suicidal	ML specialists	0.45
	Non-specialists	0.41

We tested three aggregation methods on the non-specialists' annotations:

- Full agreement–cases where all three annotators agreed.
- Standard majority voting–a text receives a class label if at least two annotators assigned it. For multi-class cases, the label sets must fully match between at least two annotators.
- Soft majority voting–a text receives labels that appear at least twice in the combined list of annotations. Multi-class labels are split into individual classes for counting.

After each aggregation method was applied, we selected 30 examples randomly and checked for compliance with the class table. Agreement was measured as the percentage of examples where the label was deemed correct by the guideline author. For aggregation methods we also report coverage, the percentage of non-empty class assignments. The results were as follows:

- Non-specialists: 95%, 75%, 55% agreement.
- Full agreement aggregation: 100% agreement, 33% coverage.
- Standard majority voting: 91% agreement, 71% coverage.
- Soft majority voting: 91% agreement, 93% coverage.

We see that agreement between the guideline author and the non-specialist varies notably from a well-performed level to a moderate level. The full agreement, as expected, shows full agreement with low coverage. On the other hand, we see that soft majority voting has higher coverage compared to standard majority voting, keeping the same level of agreement. So, we decided to use this variant of aggregation.

Notably, during post-annotation discussions, annotators provided justifications for their labels in many cases where they disagreed with the guideline author. This highlights that the task has a high level of subjectivity that needs to be accounted for. In Table 2 we present some examples where annotators' decisions completely mismatched.

Table 2. Examples of subjective texts. Texts are taken from the corpus. We added the comments at the time of writing.

Subjective text	The comment
У меня трясутся руки, как у последнего алкаша (My hands are shaking like a drunkard's).	Does the user really have a trouble with an alcohol or is it just a speech figure describing the fear or is it perhaps physical health issue?
Я был лишь виртуальным для неё (I was just a virtual person to her).	Some of the annotators recognized it was a reflection of relationship problems, some decided it was not significant to be it.
Вот бы каждый человек, который разрушил мою менталку оплачивал бы мне сеанс у психолога (I wish every person who ruined my mental state would pay for my therapy sessions).	Again, is it just a speech figure or does this person really have problems with mental health?

3.3 Annotation Team Formation

We used two primary channels to find annotators. These are trusted professional contacts and freelance platforms, aggregator platforms where clients post tasks and freelancers apply.

A total of eight annotators were recruited: three for the test set and five for the training set. All annotators underwent the following onboarding steps:

1. Orientation meeting: Overview of the project, tasks, working conditions, and annotation interface. Annotators could ask questions.
2. Training annotation: 100 pre-annotated examples from the internal team.
3. Feedback session: Discussion of training annotations and clarifications.
4. Rapid verification cycle: During the first iteration, every 500 annotations, an ML specialist reviewed 30 newly labeled examples and provided feedback.
5. Training annotations were manually reviewed due to task-specific complexities.

Performance was evaluated at two levels:

- Correct examples: Cases where soft majority voting with the original annotation yielded a non-empty class.

- Acceptable examples: Correct examples plus annotator labels deemed valid by ML specialists. That is because of task subjectivity.

The average correct example rate was 54.2%, and the acceptable rate was 73.5%. One freelancer, achieving an 80% acceptable rate, was assigned to the test set. During rapid verification, the average error count was 5 after the first check and 2 to 3 thereafter.

3.4 Data Sampling Schemes

Our goal was to annotate 50 thousand examples. To improve annotation efficiency and control class distribution, we split this value into 5 iterations by 10 thousand, and further, each iteration was split into 5 blocks (2 thousand examples). One block is one annotation task.

We apply different sampling schemes for each iteration. Sampling was based on pre-extracted features. The sampling includes standard selection criteria and optional ones. The former applied to each iteration in such a way that ensured the uniform distribution of basic statistics. The latter is used to influence the resulting class distribution.

The standard criteria include:

- No foreign-language texts.
- Balanced text length distribution.
- Presence of first-person pronouns.

Optional criteria include:

- Predictions from a suicidal signal detection model [29].
- Subsets of emotion model predictions [30].
- Sentiment model predictions [31].
- Keyword sets.
- Etc.

3.5 General Annotation Process

The annotation workflow for a single annotation block consisted of the following steps:

1. Data sampling: Examples are sampled from the corpus using an ML specialist-approved scheme.
2. Data distribution: Samples are uploaded to Label Studio [32] and distributed across annotators.
3. Initial screening: First-time annotators complete the Beck Depression Inventory (BDI).
4. Pre-annotation check: Annotators take an emotional well-being test.
5. Annotation: Annotators label the data.
6. Post-annotation check: Annotators retake the emotional well-being test. If it was their final task, they also complete the BDI.
7. Debriefing: ML specialists hold meetings to discuss questions, suggestions, and emotional venting–a simple psychological support method to alleviate stress.
8. Guideline adjustments: ML specialists revise the guidelines as needed.

3.6 Annotation verification

To ensure the quality of annotated data, it was necessary to develop a data verification process. It was decided to conduct verification in parallel with the primary annotation, meaning that instead of

checking the data post-annotation, the annotation of the control set was performed by the instruction creators. This approach has the advantage of saving time, as the control set annotation is conducted concurrently with the primary annotation. Given the subjective nature of the annotation, as noted earlier, it is advisable to manually review mismatched examples obtained after comparing the primary annotation and the control set to obtain a more accurate assessment. Thus, the verification algorithm is described as follows:

1. From each annotation task, a subset of data is selected to form a control set.
2. The annotation of the control set is performed by the instruction creator.
3. If necessary, make adjustments to the instructions.
4. Upon completion of the primary annotation, compare the data in the primary and control sets.
5. If the number of mismatches exceeds a predefined threshold, conduct additional verification of discrepancies between the annotator's and verifier's responses.
6. If the number of mismatches still exceeds the threshold, perform an error analysis and return the task to the annotator for revision.
7. If the number of mismatches does not exceed the threshold, the task is accepted.

Based on the review results, a meeting was held with the annotator to discuss the issues. Subsequent checks showed that no further corrective meetings were required. Since the task is subjective and involves a large number of classes, it was decided to evaluate not by specific classes but by signal direction, of which there are three: anti-suicidal, pre-suicidal, and irrelevant. It was decided to adjust the dataset based on model training results, while at this stage ensuring a clear separation into the three main signals. Evaluating at this level allows for sufficiently clear class boundaries, as anti-suicidal and pre-suicidal signals are opposites, and irrelevant signals are also clearly distinguishable simply by contrast with the target signals.

Based on the conditions, experience with data annotation, and other studies [33] involving subjective tasks, the acceptable mismatch threshold for control annotation was set at 15%. The number of examples selected from each block (consisting of 2000 examples) was 185 (9.2%). This "uneven" number was chosen during initial design when the certain degree of subjectivity was not yet known, with provisions for alternative verification methods. A total of 5 verification tasks were annotated for the test set and 25 tasks for the training set. Additionally, 2 tasks were completed to resolve annotations in the test set where soft aggregation failed to assign class labels. The volume of one task was 357 examples. The upper mismatch rate across all control tasks for the training set was 14.55%, and for the test set, it was 13.99%. Thus, both test sets met the established threshold.

3.7 The annotation correction process

After completion the annotation process, we started to analyze the whole dataset and experiment with the models. Two main problems were found:

- Based on confusion matrix of presuicidal and anti-suicidal model we figure out that the irrelevant texts heavily intertwine with significant part of the classes.
- The metrics of the anti-suicidal model was significantly lower compared to presuicidal model.

Analysis of models revealed the sources of the first problem:

- The 3rd person rule violation – the texts where the signal is not related to the author should be considered as irrelevant.

- ✓ Example: «Она сказал, что хочет поскорее сдохнуть уже» (She said she wanted to die quickly.)

- The relatively complex texts where the model is triggered by some lexical parts but actually there is no clear definition of the signal.
  - ✓ Example: «И говорить что настоящие страдания только скрытые от всех - это как говорить что те, кто реально хочет умереть уже в гробу» (And to say that real suffering is only hidden from everyone is like saying that those who really want to die are already in their coffins.)

- The texts that have multiple classes.
  - ✓ Examples: «Мне настолько всё это надоело, я не знаю, что мне делать, я постоянно думаю о смерти» (I'm so tired of all this, I don't know what to do, and I keep thinking about death.)

- Errors in the annotation.

To find the examples that are dubious, we applied the dataset cartography technique [35]. On the given map that is presented in Fig. 1, we can estimate the hard-to-learn region as a quadrant $x < 0.4$ and $y < 0.4$.

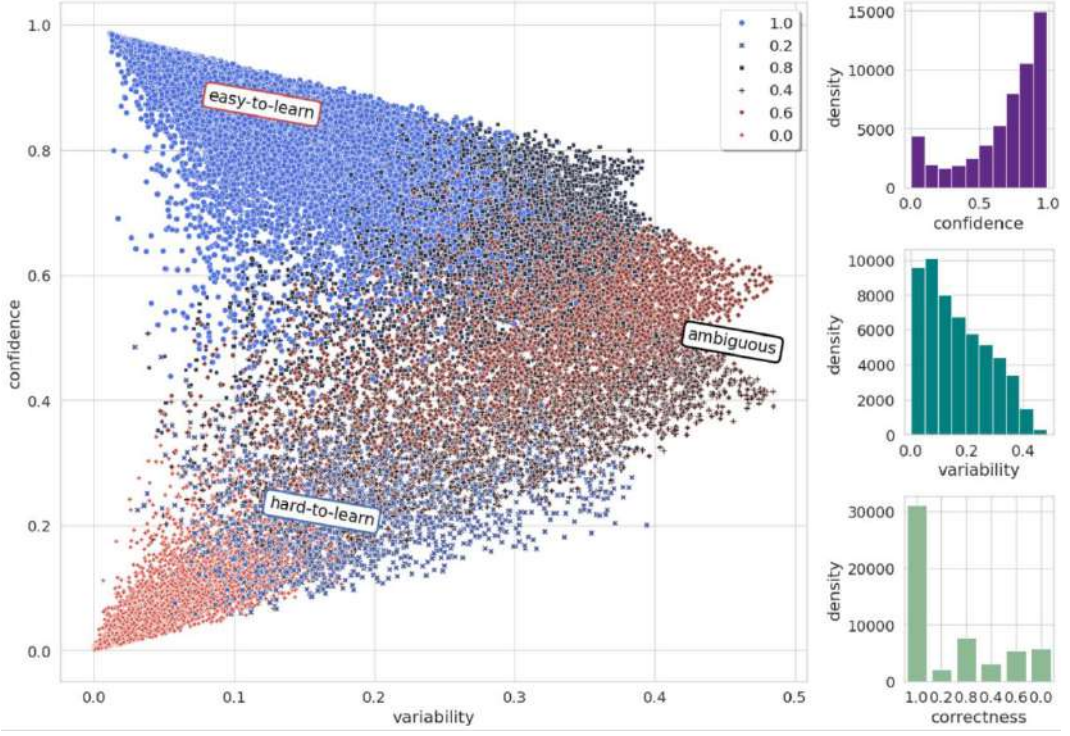


Fig. 1. Data cartography for the exact presuicidal dataset.

The count class distribution of selected examples is shown in Table 3. As we can see, the top class is an irrelevant class. Besides anti-suicidal signals that, as we know, have issues, the rest of the classes are from the feeling group. This is not surprising, as the feelings interpretation is heavily

subjective by nature. Thus, we decide to reannotate the whole irrelevant examples that don't come from the test set because the analysis shows that these examples are mostly correct but complex.

Table 3. The class distribution in hard-to-learn region.

Class name	Count
Irrelevant	1,726
Anti-suicidal signal	992
Feelings/mental emptiness, depression, longing, sadness	860
Feelings/negative self-image, guilt, shame, worthlessness, self-flagellation	763
Feelings/helplessness, hopelessness, hopelessness, despair	710

The deep analysis of the anti-suicidal model reveals that there is too much noise in the annotation that mainly comes from too abstract names of the classes. We decide to reassemble the class table of anti-suicidal classes by the next procedure:

1. 20 examples per existed class are selected.
2. The main sense of each text is annotated by ML specialist. Each specialist forms his own set of main senses.
3. The result sets are cast to the close set.
4. The two close sets are aligned to each other and grouped together.
5. The final set of groups becomes a new class table.

As an effect, several new classes are emerged saving almost all source classes but with clearer semantic meaning.

Having the new class table, we reannotated the anti-suicidal dataset. Because of the limited budget and time, the annotation was done by different schema. We decided to validate the annotators each 500 annotated examples by the batch of 50 texts. The threshold of the errors was 8 examples (15%). All checked examples become a test part (10%), while rest of the examples become train (80%) and validation parts (10%).

4. Dataset statistics

We collected 57,810 annotated examples in total. The presuicidal dataset consists of 38,406 examples, anti-suicidal dataset consists of 9702 examples, irrelevant examples are 9702 (not a mistake). Table 4 and Table 5 show the top 5 counter distribution of presuicidal and anti-suicidal dataset respectively. The final Krippendorf alpha for the test part of presuicidal dataset is 0.542. As we mentioned earlier, the significant part of the dataset is a multilable examples. The ratio of such examples is 0.261.

5. Experiments

In this section we present the results of the model learning on the final dataset. As we mentioned earlier, the structure of the classes allows us to learn the model with different levels of granularity. We hypothesize that with a high level of granularity the model will better capture the difference between classes as semantic change will be clearer. Besides the plain granularity, one might explore more complex schema where some classes are collapsed to a group and some classes rest as exact labels. It's useful if one might want to isolate some dubious classes to make the model more robust. An example of such a group is feelings.

We show results for the RuBERT [36] model only, as it constantly shows better performance compared to the RoBERTa [37] and DeBERTa [38] models.

Table 4. The class distribution of anti-suicidal dataset.

Class name	Count
Having positive social connections	1,650
Expression of love	1,384
Expression of happiness, joy, contentment	858
Positive self-assessment	595
Expression of love; Having positive social connections	486

Table 5. The class distribution of the presuicidal dataset.

Class name	Count
Death/thoughts about death	4,205
Feelings/negative self-image, guilt, shame, worthlessness, self-flagellation	3,236
Problems in the outside world/unhappy love, problems with friends, difficulties in building relationships	2,602
Feelings/helplessness, hopelessness, hopelessness, despair	2,359
Feelings/mental emptiness, depression, longing, sadness	1,964

We do basic text processing like text lowering and removing all non-alpha symbols. We also remove all multiclass examples for this experiment. The classes that have less than 100 examples were also removed. We derived the datasets from the whole data that we named as the master dataset. That means that irrelevant examples are shared across the presuicidal and anti-suicidal parts, while they actually are different, as they were sampled by different sampling schema and were annotated separately. So, the results might be biased. Table 6 shows the result performance on the different granularity levels.

As for the presuicidal model we see, that exact granularity is significantly lower than group granularity, and, on the other hand, the ternary model with irrelevant, anti-suicidal, and presuicidal classes with the binary model (relevant and irrelevant) shows even better results. That supports our hypothesis that the higher the level of classes, the better the model can distinguish them. We also see that the group presuicidal model performs on par with the anti-suicidal dataset.

Table 6. The performance of the RuBERT on different dataset settings.

Model name	Label granularity	Class count	Precision	Recall	F1-score
Presuicidal	Group	8	0.65	0.65	0.65
	Exact	26	0.61	0.51	0.53
Anti-suicidal	Exact	9	0.70	0.59	0.63
All	Binary	2	0.71	0.71	0.71
	Ternary	3	0.71	0.69	0.70

The analysis of the errors reveals that the main problem is that the irrelevant texts are notably intertwined with several classes. Manual investigation shows the same problems that were highlighted in the annotation correction section above. Especially the problem of overlapping lexical features and the 3rd rule violation.

6. Annotation instruction

Here we provide the annotation instruction translated into English along with the lists of presuicidal and anti-suicidal signals. Due to space limitations, we omit the instruction part dedicated to the

annotation interface as well as examples for each class. You can find the full version of the instruction and class tables in our repository [6].

6.1 Introduction

You will be shown texts collected from various social networks. It is necessary to distribute them according to the classification below. The annotation is needed to create a classifier that will make it easier for volunteers to find people who are on the verge of suicide so that they can be consulted and provided with assistance. Without exaggeration, we can say that by doing this work, you are contributing to saving someone's life.

6.2 How to annotate

It is necessary to study the description of pre-suicidal signals – signs in the text that indicate a possible suicide – by clicking on the [link to class table]. The description of anti-suicidal signals by clicking on the [link to class table]. The latter reduces the likelihood of suicide.

Before you start marking, you need to complete two tests.

- The emotional well-being test is completed before and after each marking block.
- The Beck Depression Scale is completed before starting all tasks and at the end of all tasks.

(The instruction of interacting with Label Studio is omitted)

6.3 Annotation principles

1. Despite data cleaning, you may encounter texts that violate the laws of the Russian Federation. We are not responsible for such texts, as they were collected from social networks. You may notify us if you come across a text that violates Russian laws via the feedback form (or messenger).
2. The content of the text must relate to the author. The text should contain first-person personal and/or possessive pronouns (I, we, my, our, me, us). For example:
    a. “I want to escape from this oppressive outside world into myself,”
    b. “Well of course I am loved, but what’s the point of being with someone who isn’t.”

    Or other speech patterns that indicate the content refers to the author. For example, “and a month later you write, ‘Kostya, I’m sorry, I met someone else, I truly love him’” – it can be said that the author received such a message. If a person talks about someone else, this information is irrelevant. Examples of irrelevant texts:
    c. “A person needs another person, or a chocolate with tea is fine too” – it cannot be determined that the author needs chocolate and tea.
    d. “Oh, Atsumu really wanted to yell at his brother, to vent his emotions and find out, to ask about all the feelings between them, and not +” – this is about a character.
    e. “Everyone has something that is most important to them.” – this refers to an abstract group of people.
    f. “She’s all dressed up with earrings and eyeshadow FOR BREAKFAST, and her husband of course looks like a water balloon that you don’t throw away for at least three years.” – this refers to third parties.
3. Messages in a foreign language should be marked as unrelated to suicide.
4. You must not attempt to interpret texts in any way, for example, assuming that the person wrote the text to attract attention. You must judge strictly according to the class descriptions. It is also not allowed to assess texts based on perceived context. Example: “but I didn’t know how or what – all my plans for the future were connected with him, in

fact everything was connected with him – favorite books, favorite music." Someone might think the author is describing negative relationships, because instinctively, if someone talks about their relationship this way, it probably ended badly. However, the text does not directly state whether the relationship ended, or whether it was bad at all.

5. If a signal is described as having happened in the past, it should also be labeled. However, for classes related to feelings, this principle applies weakly.
6. If you feel emotionally distressed or it becomes difficult for you to continue the task, you should stop immediately.
7. The texts may have unpleasant subjects; try not to annotate them right before sleep.
8. To clarify disease codes in the format F{number}.{number}, use this link to ICD-10: https://mkb-10.com/
9. If there is one clear class and it's unclear whether another class is present (especially common in short texts), do not try to look for it or spend time on it.

6.4 Specific points for dataset annotation

1. If there is an example describing a breakup, and it is not clear whether it is about a husband and wife, you should assign the class "External world problems/Unhappy love, problems with friends, difficulties building relationships"
2. If, due to some comparison, allusion, metaphor, etc., you cannot clearly determine the class, you should assign the irrelevant class. Example: "God, I must have been Commander Shepard in a past life, given how lucky I am with people in this one."

6.5 List of classes

6.5.1 List of presuisidal classes

1. Clinical manifestations/Depression – this class includes texts with facts or mentions of the diagnosis "depression", as well as symptoms of depression.
2. Clinical manifestations/Insomnia – a condition that prevents a person from falling asleep.
3. Clinical manifestations/Eating disorder (ED) – a range of behavioral syndromes associated with disruption of the eating process.
4. Clinical manifestations/Fatigue, stress, resource depletion – manifestations of physical and/or emotional fatigue, stress, and a state of exhaustion of physical and/or emotional resources.
5. Clinical manifestations/Anxiety, fear, phobias, obsessive thoughts – anxiety is a negative emotional state, expectation of trouble, impending danger, or unfavorable events, disproportionate to the actual situation, as well as sensations of fear and worry.
6. Clinical manifestations/Other mental disorders – this class includes texts mentioning any mental disorders not specified in the current list.
7. Clinical manifestations/Physical illnesses, disability – texts mentioning physical illnesses, injuries, or disability fall into this class.
8. Clinical manifestations/Past suicide attempt – this class includes texts mentioning suicide attempts in the past.
9. Clinical manifestations/Crying, hysterics – this class includes texts describing crying, tears, sobbing, hysterics, etc. (do not confuse with tearfulness).
10. Destructive behavior/Self-harm – this class reflects behaviors and thoughts related to self-harm, cutting, or inflicting physical pain on oneself.

11. Destructive behavior/Alcohol and drugs – problems and behaviors associated with alcohol, drugs, or other psychoactive substance abuse.
12. Destructive behavior/Problems with the law, discipline – facts from the past or present related to problems with the law or discipline.
13. Family problems/Bullying, physical abuse – this class includes all types of physical violence within the family and at home.
14. Family problems/Sexualized violence – events in the family environment related to sexualized violence.
15. Family problems/Family breakdown – facts related to the breakdown of the family in any form.
16. Family problems/Difficult relationships with relatives – in this class, we are interested in emotional relationship difficulties (not physical violence).
17. Family problems/Pregnancy difficulties, abortion, miscarriage – this class includes texts mentioning problems related to childbirth or pregnancy.
18. External world problems/Bullying, physical abuse – events at school, among peers, at work, or outside the home, related to bullying, emotional and physical humiliation, or violence.
19. External world problems/Sexualized violence – events at school, among peers, at work, or outside the home, related to sexualized violence.
20. External world problems/Unhappy love, problems with friends, difficulties building relationships – negative emotional states and feelings associated with situations of unhappy or unrequited love, as well as problems in friendships.
21. Death/Death, accident or suicide of loved ones – loss, death, or suicide of family members or close people, including friends.
22. Death/Thoughts about death – permissive attitudes and thoughts about suicidal behavior.
23. Death/Suicidal intentions – messages containing a concrete plan of action. Differs from "desire to die" in the declaration of actions.
24. Feelings/Helplessness, hopelessness, despair – negative emotional states, beliefs, and situations associated with feelings of helplessness and hopelessness.
25. Feelings/Negative self-perception, guilt, shame, worthlessness, self-blame – negative emotional states, beliefs, and situations associated with negative self-perception, feelings of guilt, shame, or worthlessness.
26. Feelings/Loneliness, misunderstanding, isolation, abandonment – negative emotional states, beliefs, and situations associated with feelings of loneliness, sadness, resentment, misunderstanding, etc.
27. Feelings/Aggression, anger, resentment, rage, irritability, jealousy, injustice – expression of aggression, anger, protest, rage, and other active negative emotions.
28. Feelings/Emotional suffering, emptiness, depression, melancholy, sadness – this class includes texts mentioning emotional suffering (without specifics), pain (if not in the context of physical pain), the feeling that "everything is bad", etc.
29. Other/Destructive attitudes – this class includes worldviews such as fearlessness of death, devaluation of life, reflections on the futility of life.
30. Other/Stalking – stalking is when a person is obsessively pursued. An ex-boyfriend or girlfriend, a rejected admirer, or even a stranger may wait at the entrance, watch, ambush at work, write letters, call persistently and systematically on personal or even work phones, send gifts – all these are manifestations of stalking.

31. Other/Problems with career choice and realization – this class includes texts mentioning forced dismissal, unemployment, long unsuccessful job searches, dissatisfaction with current job, or inability to find oneself in a profession.
32. Other/Money problems (poverty, debts) – temporary or permanent situations with debts, poverty, or destitution.
33. Other/Imitating an idol in suicidal behavior – this class includes texts expressing sympathy or imitation of friends, book or film characters with suicidal themes, as well as imitating an idol or public figure in suicide.

6.5.2 List of antisuisidal classes

1. Protective factors/Expression of love – this class reflects the expression of feelings of love, infatuation, and affection towards someone. This class includes texts containing such expressions.
2. Protective factors/Presence of positive social connections – this class reflects the presence of positive connections and relationships in the author's life.
3. Protective factors/Help-seeking – this class reflects the author's search for or request for help.
4. Protective factors/Desire for love and relationships – this class reflects the author's desire, need, search, or aspiration for love and warm relationships.
5. Protective factors/Positive self-esteem – this class reflects positive self-perception and high self-esteem of the author. It is the opposite of the "Negative self-perception" class from the presuicidal map.
6. Protective factors/Positive beliefs – this class reflects positive attitudes and beliefs of the author. It is the opposite of the "Negative beliefs" class from the presuicidal map.
7. Protective factors/Pursuit of happiness and joy – this class reflects the desire for, search for, and aspiration to happiness and joy.
8. Protective factors/Material desires – this class reflects material needs and desires.
9. Protective factors/Presence of goals, plans for the future, favorite activity – this class reflects the presence of goals, plans for the future, favorite activities, and is associated with some kind of activity and/or actions.
10. Protective factors/Expression of happiness, joy, satisfaction – this class reflects feelings of happiness, joy, and satisfaction.
11. Protective factors/Positive dynamics – this class reflects positive dynamics and progress in the author's emotional and/or physical state.
12. Protective factors/Supportive states and situations – this class reflects supportive states, rituals, actions, and skills of the author.

7. Conclusion

In this work we show the developed methodology of creating the large-scale dataset for detecting presuicidal and anti-suicidal signals in social media texts. By using this methodology, we collect the dataset of more than 50 thousand examples. Our experiments, despite being basic, show a promising performance level of classification models. We also find typical problems raised during the annotation and model training.

In future work we plan to deal with the annotation problem of interfering irrelevant class with relevant ones. We also want to investigate the subjectivity feature of the dataset because it significantly influences the annotation process, leading to confused annotation. Also, we would like

to explore the power of LLM for classification and annotation purposes. Furthermore, the augmentation strategies can be explored to give the model more examples of the 3rd rule. The same goes for overlapping lexica.

References

[1]. Dévora Kestel and Mark van Ommeren et al. Suicide in the world. World Health Organization, 2019. Vol. 1.
[2]. Suicide and its prevention in Russia, 2019: general facts // Demoscope URL: https://www.demoscope.ru/weekly/2020/0869/suicide.php (accessed: 18.05.2025).
[3]. Bollen J. et al. Historical language records reveal a surge of cognitive distortions in recent decades. Proc Natl Acad Sci USA, 2021. Vol. 1.
[4]. Craig J. Bryan and M. David Rudd, Brief Cognitive-Behavioral Therapy for Suicide Prevention. Guilford Press, 2018. Vol. 1.
[5]. Popov U. V., A.A. Pichikov, Suicidal behavior in adolescents. [Suicidalnoe povedenie u podrostkov] SpecLit, 2017. Vol. 1.
[6]. Kitoboy // Github URL: https://github.com/psytechlab/kitoboy (accessed: 18.05.2025).
[7]. Glen Coppersmith et al. From ADHD to SAD: Analyzing the Language of Mental Health on Twitter through Self-Reported Diagnoses // Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality, Denver, Colorado, 2015, pp. 1-10.
[8]. De Choudhury M. et al. Discovering Shifts to Suicidal Ideation from Mental Health Content in Social Media // Proceedings of the SIGCHI conference on human factors in computing systems, 2016, pp. 2098-2110.
[9]. Glen Coppersmith et al. CLPsych 2015 Shared Task: Depression and PTSD on Twitter // Proceedings of the 2 nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality. Denver, Colorado, 2015. pp. 31-39.
[10]. Losada D.E., Crestani F., A Test Collection for Research on Depression and Language Use. – Springer, Cham, 2016. Vol. 9822.
[11]. Sean MacAvaney et al. Community-level Research on Suicidality Prediction in a Secure Environment: Overview of the CLPsych 2021 Shared Task // Proceedings of the Seventh Workshop on Computational Linguistics and Clinical Psychology. Online, 2021, pp. 70-80.
[12]. Reading List for Mental Health Detection and Analysis on Social Media // Github URL: https://github.com/drmuskangarg/mentalhealthcare (accessed: 18.05.2025).
[13]. H. Andrew Schwartz et al. Personality, Gender, and Age in the Language of Social Media: The Open-Vocabulary Approach – PloS one, 2013, vol. 8
[14]. PsyEval: A Suite of Mental Health Related Tasks for Evaluating Large Language Models // ArXiv URL: https://arxiv.org/abs/2311.09189 (accessed: 18.05.2025).
[15]. Narynov S. et al. Dataset of depressive posts in Russian language collected from social media // Data in Brief, 2020, vol. 29.
[16]. Stankevich M., Smirnov I. et al. Predicting Depression from Essays in Russian // Proceedings of "Computational Linguistics and Intellectual Technologies" DIALOGUE, 2019, pp. 637-647.
[17]. Литвинова Т.А., Литвинова О.А. Языковые особенности русскоязычных текстов лиц, совершивших суицид, и лиц с высоким риском аутоагрессивного поведения // Studia Humanitatis. - 2017. № 4 / Litvinova T. A., Litvinova O. A. Linguistic features of Russian-language texts of people who have committed suicide and those at high risk of auto-aggressive behavior // Studia Humanitatis. 2017. No. 4.
[18]. Igor Buyanov and Ilya Sochenkov, The dataset for presuicidal signals detection in text and its analysis // Computational Linguistics and Intellectual Technologies. 2022. No. 21, pp. 81-92.
[19]. VK // VK URL: https://vk.com/ (accessed: 18.05.2025).
[20]. X (Twitter) // X URL: https://x.com/ (accessed: 18.05.2025).
[21]. Suicide Forum // Suicide Forum URL: http://www.suicide-forum.com/ (accessed: 18.05.2025).
[22]. A. Aluoja, J. Shlik, V. Vasar, K. Luuk, M. Leinsalu, The Emotional Well-being Questionnaire (EEK). 1999.
[23]. Тарабрина Н. В. Практикум по психологии посттравматического стресса. 1 изд., СПб.: Питер, 2001. 272 с. / Tatabatina N. V. A workshop on the psychology of post-traumatic stress. 1 edition, SPb.: Piter, 2001, 272 p.

[24]. Пакулина С.А. Психодиагностика суицидального поведения детей и подростков. 1 изд., Челябинск: 2014 / Pakulina S. A. Psychodiagnostics of suicidal behavior in children and adolescents. 1 edition, Chelabinsk: 2014.
[25]. Брайан К.Дж., Радд М.Д. Когнитивно-поведенческая терапия для предотвращения суицида. 1 изд., Москва: Вильямс, 2021. 464 с. / Brayan K. J. Radd M. D Cognitive-behavioral therapy for suicide prevention, 1 edition, Moscow: Viliams, 2021. 464 p.
[26]. Krippendorff K. Computing Krippendorff's Alpha-Reliability // 2011.
[27]. Passonneau R. Measuring Agreement on Set-valued Items (MASI) for Semantic and Pragmatic Annotation // International Conference on Language Resources and Evaluation. 2006.
[28]. Bird S., Klein E., Loper E. Natural Language Processing with Python. 1 edition. O'Reilly, 2009.
[29]. Astromis Presuicidal RuBERT // Astromis HF URL: https://hf.global-rail.com/astromis/presuisidal_rubert (accessed: 18.05.2025).
[30]. RuBERT-Tiny2 Russian Emotion Detection // Hugging Face URL: https://huggingface.co/Djacon/rubert-tiny2-russian-emotion-detection (accessed: 18.05.2025).
[31]. Blanchefort RuBERT Base Cased Sentiment // Blanchefort HF URL: https://hf.global-rail.com/blanchefort/rubert-base-cased-sentiment (accessed: 18.05.2025).
[32]. Label Studio // Github URL: https://github.com/HumanSignal/label-studio (accessed: 18.05.2025).
[33]. Sboev A., Naumov A., Rybka R. Data-Driven Model for Emotion Detection in Russian Texts // BICA*AI. 2020.
[34]. Rogers A., Romanov A., Rumshisky A., Volkova S., Gronas M., Gribov A. RuSentiment: An Enriched Sentiment Analysis Dataset for Social Media in Russian // International Conference on Computational Linguistics. 2018.
[35]. Dataset Cartography: Mapping and Diagnosing Datasets with Training Dynamics // ArXiv URL: https://arxiv.org/abs/2009.10795 (accessed: 18.05.2025).
[36]. Adaptation of Deep Bidirectional Multilingual Transformers for Russian Language // ArXiv URL: https://arxiv.org/abs/1905.07213 (accessed: 18.05.2025).
[37]. RoBERTa: A Robustly Optimized BERT Pretraining Approach // ArXiv URL: https://arxiv.org/abs/1907.11692 (accessed: 18.05.2025).
[38]. DeBERTa: Decoding-enhanced BERT with Disentangled Attention // ArXiv URL: https://arxiv.org/abs/2006.03654 (accessed: 18.05.2025).

Информация об авторах / Information about authors

Игорь Олегович БУЯНОВ – аспирант ФИЦ ИУ РАН, старший разработчик в MTS AI. Сфера научных интересов: обработка естественного языка, анализ пространств эмбеддингов, вычислительная психология.

Igor Olegovich BUYANOV – post graduate student at FRC CSC RAS, senior developer at MTS AI. Research interests: natural language processing, embedding space analysis, computational psychology.

Дарья Валентиновна ЯСЬКОВА – магистр психологии ННГУ им. Н.И. Лобачевского с 2018 года, старший разработчик в МТС ИИ с 2019 года. Сфера научных интересов: обработка естественного языка, распознавание именованных сущностей в специфичных доменах, методы аугментаций для текстовых данных.

Darya Valentinovna YASKOVA – master of psychology in N. I. Lobachevsky State University of Nizhny Novgorod from 2018, senior developer at MTS AI since 2019. Research interests: natural language processing, named entity recognition in specific domains, text augmentation methods.

Данил Сергеевич СЕРЕНКО является студентом кафедры математического моделирования и искусственного интеллекта РУДН имени Патриса Лумумбы, научным сотрудником Федерального исследовательского центра "Информатика и управление" Российской академии наук (ФИЦ ИУ РАН). Область научных интересов – искусственный интеллект, информационный поиск.

Danil Sergeevich SERENKO is a student at the Department of Mathematical Modeling and Artificial Intelligence of the Patrice Lumumba RUDN University, a researcher at Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences. His research interests include AI, information retrieval.

Данил Николаевич ШКЕРЕДА – студент Российского государственного университета нефти и газа (национальный исследовательский университет) имени И. М. Губкина, научный сотрудником Федерального исследовательского центра "Информатика и управление" Российской академии наук (ФИЦ ИУ РАН). Сфера научных интересов: эффективное обучение больших языковых моделей, семантический анализ текстов.

Danil Nikolaevich SHKEREDA is a student at the Department of Mathematical Modeling and Artificial Intelligence of the Patrice Lumumba RUDN University, a researcher at Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences. Research interests: effective training of large language models, semantic analysis of texts.

Андрей Дмитриевич ЯСЬКОВ – магистр информационных систем и технологий НГТУ им. Р. Е. Алексеева с 2017 года, разработчик в Яндекс с 2022 года. Сфера профессиональных интересов: разработка высоконагруженных веб-приложений, архитектура информационных систем, разработка интерактивных редакторов диаграмм, векторная графика, доступность веб-приложений.

Andrey Dmitrievich YASKOV – master of informatics in Nizhny Novgorod State Technical University n.a. R.E. Alekseev from 2017, developer at Yandex from 2022. Professional interests: high load web-application development, architecture of information systems, interactive editor of diagram development, vector graphics, web-application accessibility.

Илья Владимирович СОЧЕНКОВ – кандидат физико-математически наук, ведущий научный сотрудник ФИЦ ИУ РАН, ведущий научный сотрудник ИСП РАН, ведущий научный сотрудник ИППИ РАН. Сфера научных интересов: обработка естественного языка, методы информационного поиска, обработка больших массивов текстовой информации.

Ilia Vladimirovich SOCHENKOV – Cand. Sci. (Phys.-Math.), lead researcher at FRC CSC RAS, leading researcher at ISP RAS, leading researcher at IITP RAS. Research interests: Natural Language Processing, Information Retrieval, Big Data & Text Mining.

DOI: 10.15514/ISPRAS-2025-37(6)-30

Оптимизация выравнивания коротких прочтений с инделями при полногеномном секвенировании

Н.А. Колтунов, ORCID: 0009-0007-8507-8518 <n_koltunov@ispras.ru>
Е.П. Гугучкин, ORCID: 0000-0001-7885-9892 <guguchkin@ispras.ru>
Е.А. Карпулевич, ORCID: 0000-0002-6771-2163 <karpulevich@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. Представлен новый метод выравнивания прочтений для задач полногеномного секвенирования (WGS), ориентированный на повышение точности и практической эффективности этого этапа геномного анализа. В отличие от графовых подходов, предложенный алгоритм интегрирует информацию об известных генетических вариантах напрямую в процесс выравнивания, что позволяет улучшить сопоставление последовательностей с эталонным геномом без строительства сложных графовых структур. Метод продемонстрировал высокую эффективность на реальных данных: наблюдается устойчивый прирост качества выравнивания на участках с высоким уровнем изменений между разными людьми, а также участках, которые сложны для однозначного выравнивания даже при отсутствии изменений в этом месте у конкретного человека. В частности, использование информации о вариантах позволяет точнее выравнивать короткие последовательности (прочтения), содержащие альтернативные аллели, снижая число ошибок в указанных регионах. При этом требуемые вычислительные ресурсы остаются на приемлемом уровне, что делает решение применимым в стандартных WGS-пайплайнах без существенного увеличения нагрузки. Скорость работы алгоритма сопоставима с традиционными решениями, что упрощает его интеграцию в существующие аналитические программные конвейеры. Практическая ценность метода заключается в улучшении точности выравнивания, что напрямую влияет на качество последующего обнаружения вариантов и других анализов. Предлагаемый подход способен служить эффективной альтернативой современным графовым методам выравнивания, обеспечивая сопоставимое повышение качества результатов выравнивания при меньшей сложности реализации. Перспективы дальнейшего развития включают оптимизацию производительности алгоритма, расширение набора учитываемых генетических вариантов и проведение углубленного сравнения с другими инструментами. Эти шаги призваны еще более повысить эффективность и надежность метода, укрепляя его значимость для практического применения в геномике.

Ключевые слова: выравнивание коротких прочтений; инделы; альтернативные контиги; преобразование координат; определение вариантов.

Для цитирования: Колтунов Н.А., Гугучкин Е.П., Карпулевич Е.А. Оптимизация выравнивания коротких прочтений с инделями при полногеномном секвенировании. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 211–222. DOI: 10.15514/ISPRAS–2025–37(6)–30.

Благодарности: Исследование выполнено при поддержке Российского научного фонда, проект № 25-21-20111.

Optimization of Short Reads Alignment with Indels in Whole-Genome Sequencing

N.A. Koltunov, ORCID: 0009-0007-8507-8518 <n_koltunov@ispras.ru>
E.P. Guguchkin, ORCID: 0000-0001-7885-9892 <guguchkin@ispras.ru>
E.A. Karpulevich, ORCID: 0000-0002-6771-2163 <karpulevich@ispras.ru>
Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. We present a novel method for aligning reads in whole-genome sequencing (WGS), aimed at improving alignment accuracy and the practical efficiency of this stage of genomic analysis. Unlike graph-based approaches, the proposed algorithm directly integrates knowledge of known genetic variants into the alignment process, enabling more accurate mapping of reads to the reference genome without constructing complex graph structures. The method has demonstrated high effectiveness on real sequencing data: we observed a consistent improvement in read alignment quality in highly variable and difficult-to-map regions of the genome. In particular, using variant information allows more precise alignment of reads that contain alternative alleles, reducing the number of mapping errors in these regions. At the same time, the required computational resources remain at an acceptable level, making this solution applicable in standard WGS pipelines without a significant increase in workload. The alignment speed of the algorithm is comparable to traditional solutions, which facilitates its integration into existing analytical pipelines. The practical value of the method lies in the improved alignment accuracy, which directly affects the quality of downstream variant calling and other analyses. The proposed approach can serve as an effective alternative to current graph-based alignment methods, providing comparable improvements in alignment quality with lower complexity of implementation. Future work will include optimizing the algorithm's performance, expanding the set of genetic variants accounted for, and conducting in-depth comparisons with other tools. These steps are intended to further increase the method's efficiency and reliability, reinforcing its significance for practical use in genomics.

Keywords: short-read alignment; indels; alternate contigs; liftover; variant calling.

For citation: Koltunov N.A., Guguchkin E.P., Karpulevich E.A. Optimization of short reads alignment with indels in whole-genome sequencing. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 2, part 2, 2025, pp. 211-222 (in Russian). DOI: 10.15514/ISPRAS-2025-37(6)-30.

Acknowledgements. This research was funded by the Russian Science Foundation (RSF) project No 25-21-20111.

1. Введение

Современные технологии секвенирования геномов становятся все дешевле и доступнее, что приводит к переходу от секвенирования отдельных геномов к массовому секвенированию популяций. Ярким примером является проект UK Biobank по секвенированию геномов жителей Великобритании, в рамках которого уже получены сотни тысяч полных геномов вместе с клиническими данными [1]. Накопление данных массового секвенирования породило обширные базы известных генетических вариантов человека, такие как база Human Pangenome Reference Consortium (HPRC) [2], содержащая сотни различных гаплотипов для более полного охвата генетического разнообразия. К ним примыкает и проект 1000 Genomes [3], ставший первой масштабной инициативой по систематическому каталогу распространенных вариантов в разных популяциях и заложивший методологические и ресурсные основы для последующих пангеномных эталонов. Использование информации о ранее известных вариантах генома конкретной популяции открывает возможности для улучшения биоинформатических методов анализа, в частности методов выравнивания прочтений и поиска вариантов.

Полное секвенирование генома (Whole Genome Sequencing, WGS) с применением технологий Next-Generation Sequencing (NGS) генерирует огромные объемы коротких фрагментов ДНК

– коротких прочтений, обычно длиной порядка 150 пар оснований. Выравнивание этих прочтений на референсный геном является одним из самых главных этапов анализа геномных данных. Широко используется двухшаговый подход (seed-and-extend), при котором и референс, и прочтения разбиваются на короткие подпоследовательности (сиды), ищутся совпадающие якорные участки, выстраиваются цепочки якорей, а затем выполняется точное выравнивание алгоритмом, например, Смита–Ватермана или Нидлмана–Вунша. Популярный инструмент выравнивания minimap2 [4] реализует близкую схему (seed-chain-align): после поиска сидов строится цепочка коллинеарных якорей (anchors), и уже вдоль нее выполняется точное выравнивание; при этом референс индексируется по минимимайзерам с использованием хеш-структуры, сопоставляющей минимайзеры их позициям в референсе. Благодаря минимайзерам такой индекс позволяет быстро находить соответствия между прочтениями и референсом в масштабах всего генома.

Однако наличие генетических вариантов в секвенируемом геноме затрудняет выравнивание коротких прочтений на линейный референс. Если прочтений содержит индель (вставку или делецию относительно референса) или другой вариант, качество его выравнивания снижается – метрика MAPQ заметно падает. В ряде случаев такие прочтений вовсе не выравниваются в верное положение, что приводит к потерянным (ложно-негативным) вариантам при последующем вызове вариантов. Известно, что более двух третей структурных вариантов (SV) не обнаруживались при анализе данных коротких прочтений именно из-за отсутствия альтернативных аллелей в используемом референсном геноме. Проблемы возникают и в сложных регионах генома (например, повторах), где стандартные алгоритмы выравнивания коротких прочтений работают недостаточно точно [5]. Все это описывает проявление смещения относительно референса, при котором последовательности, отличающиеся от референса, выравниваются хуже.

Одним из путей решения проблемы является переход от линейного референса к пангеномному референсу или графу генома, учитывающему известные варианты. Построение графовой модели референса (genome graph) позволяет представить альтернативные аллели и индели в виде альтернативных путей и узлов графа. Графовые выравниватели (например, VG Giraffe [6]) демонстрируют более высокую точность в переменных регионах генома по сравнению с выравниванием на линейный эталон. Так, консорциум HPRC недавно представил первый черновой пангеномный референс человека, включающий 47 фазированных диплоидных сборок генома, добавив около 119 миллионов пар оснований новых последовательностей и значительно повысив выявление структурных вариантов. Использование этого пангенома при анализе коротких прочтений позволило сократить ошибки обнаружения вариантов и увеличить число выявляемых структурных вариантов по сравнению с традиционным процессом на основе GRCh38. Однако полные графовые решения остаются сложными вычислительно: построение и хранение индекса пангенома требует больших объемов памяти и времени. Это стимулирует поиск компромиссных подходов, сочетающих скорость линейного выравнивания с учетом известных вариантов.

Один из таких подходов – использование расширенного линейного референса с альтернативными контигами, содержащими известные варианты. Еще проект GRCh38 включал альтернативные контиги для сложных регионов, а современные методы развивают эту идею на уровне популяционных данных. Например, в проприетарном инструменте Illumina DRAGEN реализована мультигеномная карта: к основному референсу добавлены дополнительные последовательности – фрагменты гаплотипов из разных популяций [7]. Прочтения выравниваются одновременно на основной референс и на эти альтернативные контиги, связанные с ним через заранее рассчитанные соответствия (лифтовер-цепочки). Далее выравнивания оцениваются группами: если прочтение лучше выровнялся на альтернативный контиг, это выравнивание становится основным, но позиция прочтений все равно переводится на координаты основного референса, и в итоговом SAM/BAM файле

указывается стандартный хромосомный референс. Таким образом, внешне процесс не отличается от классического (все координаты в привычной сборке), а внутренняя точность выравнивания повышается за счет использования информации о популяционных вариантах. Показано, что такой ALT-ориентированный подход заметно улучшает точность выравнивания в сложных областях и снижает число ошибок в вызове SNP и инделов.

В данной работе предложен новый метод улучшения выравнивания коротких прочтений за счет модификации референсной последовательности для инструмента выравнивания minimap2. В отличие от полноценных графовых решений, наш подход сохраняет линейную структуру референса, но дополняет его контигами с инделями, отражающими известные варианты. По сути, референс обогащается альтернативными путями для прочтений, содержащих вставки или делеции, характерные для данной популяции. После индексирования такой расширенной базы в minimap2, прочтения выравниваются стандартным образом – наиболее подходящие кандидаты для выравнивания будут найдены даже если они содержат варианты, представленные на добавленных контигах. Инструмент LevioSAM2 [8] затем используется для преобразования координат (лифтовера) полученных выравниваний: все позиции прочтений, выровненных на добавленные контиги, автоматически пересчитываются в координаты оригинального референса. В результате выравнивания и последующий поиск вариантов осуществляются относительно стандартной последовательности, но с повышенной точностью. Предлагаемый подход позволяет уменьшить число ложно не обнаруженных генетических вариантов (особенно инделей) и повысить качество выравнивания без существенного увеличения вычислительной нагрузки. Таким образом, наш метод может рассматриваться как альтернатива графовым методикам при анализе геномных данных человека (и потенциально других видов), сочетая преимущества учета известных вариаций генома с простотой и быстродействием линейного выравнивания.

2. Материалы и методы исследования

2.1 Выравнивание коротких прочтений с помощью minimap2

Для выравнивания прочтений применялся инструмент minimap2 (версия 2.30). Он реализует стандартный алгоритм типа seed–chain–align, широко используемый в полногеномных инструментах выравнивания. На этапе индексирования референсный геном разбивается на короткие k-меры, из которых отбираются минимайзеры – наиболее лексикографически малые подстроки в скользящем окне фиксированной длины. Minimap2 собирает такие минимайзеры референса и сохраняет их в хеш-индексе: ключом служит хеш минимайзера, а значением – конкатенированный список позиций данного минимайзера в геноме. За счет минимайзеров размер индекса существенно уменьшается без потери чувствительности.

Далее каждое входное чтение обрабатывается следующим образом. Minimap2 вычисляет минимайзеры и для прочтения, затем с помощью индекса быстро находит все совпадающие позиции этих минимайзеров на референсе. Эти совпадения служат якорями (anchors) для предварительного выравнивания. На основе набора якорных совпадений выполняется поиск максимально длинной цепочки коллинеарных якорей – то есть последовательности совпадений, расположенных на прочтении и в геноме в одном и том же порядке (без нарушения относительного расположения). Это позволяет определить примерное положение сегмента прочтения на геноме, даже если между якорями имеются разрывы из-за инделов или других вариантов. Для найденной цепочки minimap2 выполняет точное выравнивание прочтения на соответствующий участок референса, используя алгоритм динамического программирования (реализован в библиотеке KSW2). В результате формируется окончательное выравнивание с CIGAR-строкой, отражающей совпадающие и несовпадающие базы, вставки и делеции. Если для одного прочтения обнаруживается

несколько почти равных по качеству цепочек, инструмент выводит альтернативные (вторичные) выравнивания с пониженным MAPQ. Однако в общем случае minimap2 выбирает одно наилучшее (первичное) выравнивание для каждого прочтения.

Minimap2 изначально разработан для длинных чтений, но включает и режим для коротких прочтений (параметры -x sr и параметры -k27 -w14), который были использованы в наших экспериментах. В этом режиме параметры индексирования и выравнивания адаптированы под длину чтений ~150 пар оснований (“букв” из множества A, C, T, G), что соответствует типичным данным секвенирования платформ Illumina (парные прочтения 2×150 пар оснований). Все прочтения в тестовых наборах данных имели стандартную длину 150 пар оснований и выравнивались на референсный геном человека (сборка GRCh38) с использованием описанного алгоритма. Для оценки нового подхода мы сравнивали качество выравнивания minimap2 в обычном режиме (линейный референс) и с учетом дополнительных последовательностей, как описано ниже.

2.2 Выравнивание коротких прочтений с помощью minimap2

В рамках предлагаемого подхода референсная последовательность была расширена за счет добавления альтернативных контигов, содержащих известные вариантные аллели. В качестве источника вариантов использована информация пангеномной базы HPRC (Human Pangenome Reference Consortium), в которой накоплены различные аллели, отсутствующие в линейном референсе GRCh38. Мы сконцентрировались на инсерциях и делециях (инделях) – вариантах типа вставок/делеций, которые особенно затрудняют выравнивание коротких прочтений на линейный эталон. Каждый такой вариант добавлялся в референс в виде отдельного контига: из последовательности основного референса выделялся небольшой фланкирующий участок вокруг позиции варианта, и в него либо вставлялась дополнительная последовательность (для инсерций), либо удалялся фрагмент (для делеций), таким образом формируя альтернативный аллель. Полученный контиг представляет собой альтернативный путь длиной в несколько сотен пар оснований, включающий данный вариант в контексте исходного генома (рис. 1).

Все сгенерированные альтернативные контиги были добавлены в референсный FASTA-файл как отдельные последовательности (аналогично тому, как альтернативные аллели представлены в сборке GRCh38). В названиях контигов указывалась родительская хромосома и идентификатор варианта, что позволяло отличить их от основных хромосом. После этого расширенный референс (включающий стандартные хромосомы и дополнительные контиги) индексировался инструментом minimap2 аналогично обычному геному. Выравнивание прочтений выполнялось одновременно на весь этот комбинированный референс. В результате для прочтений, содержащих известные индели, minimap2 мог находить выравнивания не только на основном эталоне, но и на добавленных альтернативных последовательностях.

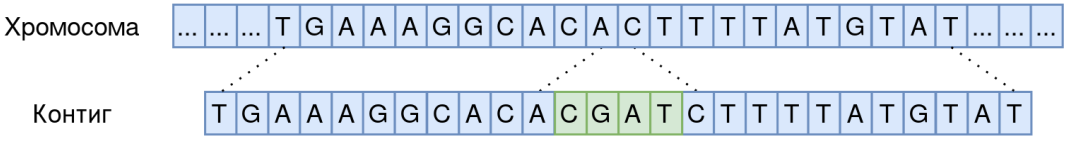


Рис. 1. Пример созданного контига с добавленной инсерцией (A → ACGAT).
Fig. 1. Example of a constructed contig with an added insertion (A → ACGAT).

Чтобы корректно обработать случаи множественного выравнивания, мы реализовали пост-обработку результатов выравнивания перед этапом лифтовера координат. Minimap2 с расширенным референсом нередко выдает два выравнивания для одного прочтения: одно – на основной хромосоме (с пропуском или разрывом в месте варианта), и второе – на альтернативном контиге, содержащем вставку или делецию. Мы сгруппировали такие

выравнивания по идентификатору прочтения и сравнивали качество (баллы выравнивания, MAPQ). Если альтернативный контиг давал более качественное выравнивание (меньше расхождений, выше выровненный процент и MAPQ), то выравнивание на основном референсе считалось субоптимальным и удалялось из файла. И наоборот, если прочтение лучше выровнялось на линейный референс, дополнительное выравнивание на контиг считалось избыточным. Таким образом, для каждого прочтения оставалось не более одного наилучшего выравнивания – фактически мы выбирали между референсным и альтернативным путем для прочтений с вариантом. Данный прием предотвращает снижение суммарной метрики MAPQ из-за наличия двух конкурирующих позиций выравнивания для одного и того же фрагмента.

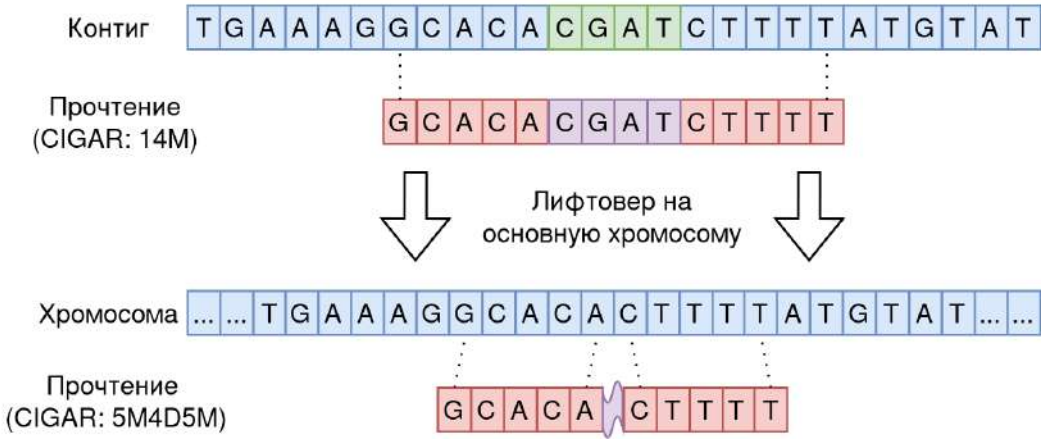


Рис. 2. Пример переноса выровненного прочтения на основную хромосому.
Прочтение содержащее инсерцию A -> ACGAT выровнялось на созданный контиг.
Строка CIGAR 14M означает, что всё прочтение, состоящее из 14 п.о. выровнялось без вставок и делеций. После лифтовера прочтения на основную хромосому строка CIGAR стала 5M4D5M, что означает: 5 п.о. без вставок и делеций, 4 п.о. – делеция и снова 5 п.о. без вставок и делеций.
Fig. 2. Example of lifting over an aligned read to the primary chromosome.
The read containing the insertion A → ACGAT aligned to the constructed contig.
The CIGAR string 14M indicates that the entire read of 14 bp aligned with no insertions or deletions.
After liftover of the read to the primary chromosome, the CIGAR string became 5M4D5M, which means: 5 bp matching (no indels), a 4-bp deletion, and then another 5 bp matching (no indels).

После отбора для каждого прочтения единственного наилучшего выравнивания (референсного или альтернативного), следующим шагом было перевести эти выравнивания на координаты основного эталонного генома. Для этого мы воспользовались инструментом LevioSAM2, специально разработанным для высокоточного переноса выравниваний между различными версиями или сборками генома. В нашем случае инструмент LevioSAM2 применял подготовленный нами chain-файл (файл цепочек), описывающий соответствие между последовательностями добавленных альтернативных контигов и участком основной хромосомы, чтобы корректно пересчитать координаты выравниваний. Важно отметить, что инструмент LevioSAM2 не просто смещает координаты: он обновляет всю информацию о выравнивании (название референса, позицию, флаги, CIGAR-строку и т.д.) в соответствии с целевым референсом. В результате выравнивания прочтений, изначально полученные на дополнительных контигах, были преобразованы в эквивалентные выравнивания на линейном референсном геноме (рис 2.).

Хотя LevioSAM2 выполняет перенос координат достаточно точно, в некоторых случаях оставались мелкие неточности выравнивания. Например, если прочтение содержало индель немного отличающийся от представленного на контиге, то после лифтовера его CIGAR-

строка могла сместиться на пару позиций. Для дополнительной корректировки мы применили инструмент локального перевыравнивания ABRA2 (версия 2.24). ABRA2 выполняет локальную пересборку участков с предположительными инделями, что позволяет устранить систематические ошибки выравнивания вокруг инсерций/делеций. В контексте нашего подхода ABRA2 донастраивал выравнивания прочтений в тех местах, где лифтовер мог быть неточным, тем самым выравнивания окончательно приводились в согласованное состояние относительно основного референса.

После этапов лифтовера и перевыравнивания полученные выравнивания проходят стандартную обработку: с помощью samtools выполнялась сортировка по координатам, маркировка дубликатов PCR, и полученный BAM-файл выравнивания поступал в инструмент определения вариантов DeepVariant. Отметим, что за счет обратного переноса координат весь последующий анализ вариаций осуществлялся в системе координат GRCh38, как и в классическом подходе.

3. Результаты

Все эксперименты проводились с использованием эталонной человеческой геномной сборки HG002, взятой из проекта Genome in a Bottle (GIAB) [10], а также парные прочтения из FDA Precision Truth Challenge V2 [11]. Запуски осуществлялись с помощью конвейера nf-core/sarek (v3.4.0). В качестве источника вариантов для создания контигов была использована информация пангеномной базы HPRC, из которой были извлечены индели с необходимыми свойствами. Все запуски сравнивались с эталоном с помощью инструмента hap-py [12]. При эталонном запуске (таблица 1) на этапе выравнивания был применен minimap2 (v2.30 с параметрами -x sr -k 27 -w 14), на этапе вызова вариантов был использован DeepVariant (v1.9.0).

Табл. 1. Эталонный запуск на оригинальной последовательности GRCh38.
Table 1. Reference run on the original GRCh38 sequence.

Type	TOTAL	TP	FN	FP	Recall	Precision	F1_Score
INDEL	525469	522364	3105	1222	0.994091	0.997666	0.995875
SNP	3365127	3348351	16776	4444	0.995015	0.998675	0.996841

После модификации алгоритма выравнивания и создания дополненных геномных референсных последовательностей потребовалось перебрать большое количество комбинаций различных параметров для поиска оптимальных результатов. Первый из важных параметров – размер фланкирующего участка при создании контигов (расстояние влево и вправо от добавленного инделя), значения которого при экспериментах равнялись 100, 150, 200, 300, 500. Брать длину фланкирующего участка меньше 100 не имеет смысла, так как тогда короткое прочтение длиной 150 не сможет на него выровняться, а длина фланкирующего участка больше 500 создает слишком много «шума» на границах этого контига и увеличивает количество ложноположительных вариантов. Также при создании контигов можно учитывать длину самих инделей, так как очень длинные (или очень короткие) могут негативно влиять на результаты. Верхней границей было определено значение 100 (и оно остается таковым при всех запусках, которые будут указаны далее), так как контиги со слишком большими инделями сильно увеличивают ложноположительные варианты, нижней границей взято значение 10, что также уменьшает количество неверных результатов. И еще один из основных перебираемых параметров – это значение AF (Allele Frequency) – это частота альтернативной аллели в выборке, представленная числом от 0 до 1. Она показывает, какая доля всех аллелей в популяции (или наборе образцов) приходится на альтернативную аллель. Для улучшения точности результатов мы можем повышать это значение, чтобы брать только более часто встречающиеся индели.

Были выбраны значения длины инделей от 10 до 100, при этом взялись те индели, значение AF у которых больше 0.1. В дальнейших запусках было исследовано влияние длины фланкирующего участка при создании контигов.

Табл. 2. Результаты запуска на референсе GRCh38 с добавлением контигов. Длина фланкирующего участка 100. AF > 0.1. Индели длины от 10 до 100.
Table 2. Results of the run on the GRCh38 reference with added contigs. Flanking region length: 100 bp. AF > 0.1. Indels 10–100 bp long.

Type	TOTAL	TP	FN	FP	Recall	Precision	F1-Score
INDEL	525469	522515	2954	1228	0.994378	0.997655	0.996014
SNP	3365127	3348470	16657	4412	0.99505	0.998684	0.996864

Табл. 3. Результаты запуска на референсе GRCh38 с добавлением контигов. Длина фланкирующего участка 300. AF > 0.1. Индели длины от 10 до 100.
Table 3. Results of the run on the GRCh38 reference with added contigs. Flanking region length: 300 bp. AF > 0.1. Indels 10–100 bp long.

Type	TOTAL	TP	FN	FP	Recall	Precision	F1-Score
INDEL	525469	522540	2929	1313	0.994426	0.997493	0.995957
SNP	3365127	3348469	16658	4474	0.99505	0.998666	0.996855

Табл. 4. Результаты запуска на референсе GRCh38 с добавлением контигов. Длина фланкирующего участка 500. AF > 0.1. Индели длины от 10 до 100.
Table 4. Results of the run on the GRCh38 reference with added contigs. Flanking region length: 500 bp. AF > 0.1. Indels 10–100 bp long.

Type	TOTAL	TP	FN	FP	Recall	Precision	F1-Score
INDEL	525469	522536	2933	1341	0.994418	0.99744	0.995927
SNP	3365127	3348415	16712	4621	0.995034	0.998622	0.996825

Было получено (табл. 2-4), что на SNP количество верно найденных (TP) вариантов растет при каждом уменьшении длины фланкирующего участка, а количество ложноположительных (FP) уменьшается, что дает рост метрик Recall и Precision, и, как следствие, улучшение метрики F1. На инделях количество верно найденных вариантов немного уменьшается при снижении длины фланкирующего участка, но при это все равно остается выше, чем при эталонном запуске. Количество же ложно-положительных вариантов значительно падает при уменьшении длины фланкирующего участка, что дает хороший рост метрики F1. По результатам данных экспериментов размер окна был зафиксирован на значении 100, как лучший результат.

Далее были рассмотрены варианты границы значения AF >0.05, >0.1 (табл. 2), >0.15.

Табл. 5. Результаты запуска на референсе GRCh38 с добавлением контигов. Длина фланкирующего участка 100. AF > 0.05. Индели длины от 10 до 100.
Table 5. Results of the run on the GRCh38 reference with added contigs. Flanking region length: 100 bp. AF > 0.05. Indels 10–100 bp long.

Type	TOTAL	TP	FN	FP	Recall	Precision	F1-Score
INDEL	525469	522521	2948	1242	0.99439	0.997628	0.996007
SNP	3365127	3348464	16663	4411	0.995048	0.998685	0.996863

Табл. 6. Результаты запуска на референсе GRCh38 с добавлением контигов. Длина фланкирующего участка 100. AF > 1.5. Индели длины от 10 до 100.
Table 6. Results of the run on the GRCh38 reference with added contigs. Flanking region length: 100 bp. AF > 0.15. Indels 10–100 bp long.

Type	TOTAL	TP	FN	FP	Recall	Precision	F1-Score
INDEL	525469	522496	2973	1215	0.994342	0.99768	0.996008
SNP	3365127	3348471	16656	4411	0.99505	0.998685	0.996864

По результатам (табл. 2, 5, 6) было замечено, что оптимальным по метрике F1 является значение AF > 0.1, так как отклонение в большую сторону уменьшает количество верно определенных вариантов на инделях, а отклонение в меньшую сторону увеличивает количество ложно-положительных.

Также в качестве эксперимента (табл. 7) было проверено влияние изменения параметров minimap2 (-k 21 -w 11), однако это снижает количество верно определенных вариантов на снипах ниже эталонного запуска (табл. 1), с которым мы сравнивались, поэтому было оставлено предыдущее значение (-k 27 -w 14).

Табл. 7. Результаты запуска на референсе GRCh38 с добавлением контигов. Длина фланкирующего участка 100. AF > 0.1. Индели длины от 10 до 100. Параметры minimap2: -k 21 -w 11.
Table 7. Results of the run on the GRCh38 reference with added contigs. Flanking region length: 100 bp. AF > 0.1. Indels 10–100 bp long. Minimap2 parameters: -k 21 -w 11.

Type	TOTAL	TP	FN	FP	Recall	Precision	F1-Score
INDEL	525469	522508	2961	1213	0.994365	0.997684	0.996022
SNP	3365127	3347736	17391	4446	0.994832	0.998674	0.996749

После подбора оптимальных значений AF и длины фланкирующего участка был проведен эксперимент (табл. 8) с удалением нижнего ограничения на длину инделей (>10).

Табл. 8. Результаты запуска на референсе GRCh38 с добавлением контигов. Длина фланкирующего участка 100. AF > 0.1. Индели длины до 100.
Table 8. Results of the run on the GRCh38 reference with added contigs. Flanking region length: 100 bp. AF > 0.1. Indels less then 100 bp long.

Type	TOTAL	TP	FN	FP	Recall	Precision	F1-Score
INDEL	525469	522707	2762	1334	0.994744	0.997454	0.996097
SNP	3365127	3348441	16686	4486	0.995041	0.998662	0.996849

Данный набор параметров дает наибольшее количество верно определенных вариантов на инделях и, несмотря на рост ложноположительных вариантов, самый лучший по F1-метрике результат. Однако на снипах был получен рост ложноположительных и падение верно определенных результатов, что в сумме дает результат хуже, чем на запуске из табл. 2. После проведения различных экспериментов был сделан вывод, что по F1-метрике лучший результат на инделях дают параметры из таблицы 8, а для снипов – из табл. 2.

4. Заключение

Таким образом, предложенный метод выравнивания демонстрирует высокую эффективность на практике. Он обеспечивает существенный прирост качества выравнивания прочтений по сравнению с традиционными подходами, что особенно заметно в сложных для картирования регионах генома. Достигнутые результаты показывают, что метод может служить эффективной альтернативой графовым подходам к выравниванию прочтений WGS,

обеспечивая сопоставимое улучшение точности без значительного усложнения или увеличения вычислительной нагрузки.

В дальнейшем планируется провести оптимизацию производительности алгоритма для ускорения вычислений и уменьшения потребления ресурсов. Кроме того, мы намерены расширить набор учитываемых в выравнивании генетических вариантов, чтобы еще сильнее повысить полноту и точность покрытия вариабельных участков генома. Важным шагом станет также сравнительный анализ предложенного решения с современными инструментами выравнивания, такими как VG Giraffe, DRAGEN и другими. Эти шаги позволят всесторонне оценить преимущества метода и подтвердят его конкурентоспособность как надежного и эффективного инструмента для выравнивания прочтений при полногеномном секвенировании.

Определения терминов

Гаплотипы	версия генетической последовательности, унаследованная от одного родителя
Референсная последовательность	стандартная версия генома, используемая как эталон для сравнения
Выравнивание	процесс установления соответствия между позициями коротких последовательностей (прочтений) и референсной последовательности
Генетический вариант (SNP)	замена одного нуклеотида (буквы) в определенной позиции генома
Индель	вставка или удаление участка ДНК (по первым буквам от инсерция/делеция)
Контиг	непрерывный фрагмент последовательности
Лифтовер	процесс переноса выравниваний (текстовых последовательностей) на референсный геном
Лифтовер-цепочки	соответствие между последовательностями добавленных альтернативных контигов и участком основной хромосомы для корректного пересчета координат выравниваний
Минимайзеры	наиболее лексикографически малые подстроки в скользящем окне фиксированной длины

Список литературы / References

[1]. Halldorsson, B. V., Eggertsson, H. P., Moore, K. H., Hauswedell, H., Eiriksson, O., Ulfarsson, M. O., ... & Stefansson, K. (2022). The sequences of 150,119 genomes in the UK Biobank. Nature, 607(7920), 732-740.

[2]. Liao, W. W., Asri, M., Ebler, J., Doerr, D., Haukness, M., Hickey, G., ... & Paten, B. (2023). A draft human pangenome reference. Nature, 617(7960), 312-324.

[3]. Genomes Project Consortium. A global reference for human genetic variation. Nature. 2015;526(7571):68.

[4]. Li, H. (2018). Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics, 34(18), 3094-3100.

[5]. Chaisson, M. J., Sanders, A. D., Zhao, X., Malhotra, A., Porubsky, D., Rausch, T., ... & Lee, C. (2019). Multi-platform discovery of haplotype-resolved structural variation in human genomes. Nature communications, 10(1), 1784.

[6]. Sirén, J., Monlong, J., Chang, X., Novak, A. M., Eizenga, J. M., Markello, C., ... & Paten, B. (2021). Pangenomics enables genotyping of known structural variants in 5202 diverse genomes. Science, 374(6574), abg8871.

[7]. Illumina. (n.d.). ALT-aware mapping. Illumina DRAGEN Bio-IT Platform Documentation (v3.7). Available at: https://support.illumina.com/content/dam/illumina-support/help/Illumina_DRAGEN_Bio_IT_Platform_v3_7_1000000141465/Content/SW/Informatics/Dragen/GPipelineAltMap_fDG.html, accessed 12.11.2025.
[8]. Mun, T., Chen, N. C., & Langmead, B. (2021). LevioSAM: fast lift-over of variant-aware reference alignments. Bioinformatics, 37(22), 4243-4245.
[9]. Mose, L. E., Perou, C. M., & Parker, J. S. (2019). Improved indel detection in DNA and RNA via realignment with ABRA2. Bioinformatics, 35(17), 2966-2973.
[10]. National Institute of Standards and Technology (NIST). (n.d.). Genome in a Bottle. NIST. Available at: https://www.nist.gov/programs-projects/genome-bottle, accessed 13.11.2025.
[11]. Olson, N. D., Wagner, J., McDaniel, J., Stephens, S. H., Westreich, S. T., Prasanna, A. G., ... & Zook, J. M. (2022). PrecisionFDA Truth Challenge V2: Calling variants from short and long reads in difficult-to-map regions. Cell genomics, 2(5).
[12]. Krusche, P., Trigg, L., Boutros, P. C., Mason, C. E., De La Vega, F. M., Moore, B. L., ... & Global Alliance for Genomics and Health Benchmarking Team. (2019). Best practices for benchmarking germline small-variant calls in human genomes. Nature biotechnology, 37(5), 555-560.

Информация об авторах / Information about authors

Никита Артемович КОЛТУНОВ – лаборант Федерального государственного бюджетного учреждения науки Институт системного программирования им. В.П. Иванникова Российской академии наук, специалист в области биоинформатики.

Nikita Artemovich KOLTUNOV – Laboratory assistant at the Ivannikov Institute for System Programming of the Russian Academy of Sciences, a specialist in bioinformatics.

Егор Павлович ГУГУЧКИН – аспирант Федерального государственного бюджетного учреждения науки Институт системного программирования им. В.П. Иванникова Российской академии наук, специалист в области биоинформатики.

Egor Pavlovich GUGUCHKIN – postgraduate student at the Ivannikov Institute for System Programming of the Russian Academy of Sciences, a specialist in bioinformatics.

Евгений Андреевич КАРПУЛЕВИЧ – кандидат физико-математических наук, научный сотрудник Института системного программирования им. В.П. Иванникова Российской академии наук, специалист в области биоинформатики.

Evgeny Andreevich KARPULEVICH – Cand. Sci. (Phys.-Math.), Ivannikov Institute for System Programming of the Russian Academy of Sciences.

DOI: 10.15514/ISPRAS-2025-37(6)-31

ExpressPrint: An Approach to Watermarking of Visual Foundation Models

[1,2] *A.S. Chistyakova, ORCID: 0000-0003-4896-4418 <a.chistyakova@ispras.ru>*
[1,3] *M.A. Pautov, ORCID: 0000-0003-0438-6361 <pautov@airi.net>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*

[2] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia.*

[3] *AIRI, 4th floor, building 2, 6, Presnenskaya embankment, Business complex "Empire", Moscow, 123112, Russia.*

Abstract. The substantial cost of training from scratch of visual foundation models (VFMs) on large and vast datasets motivates the models' owners to protect their intellectual property via ownership verification methods. In this work, we propose ExpressPrint, a novel approach to watermarking VFMs based on the fine-tuning of expressive layers of VFMs together with a small encoder-decoder network to embed the digital watermarks into a set of input images. Our method implies a small modification of expressive layers together with training an encoder-decoder neural network to extract user-specific binary messages from the hidden representations of certain input images. This method allows distinguishing between the foundation model provided to a user and independent models, thereby preventing unauthorized use of the model by third parties. We discover that the ability to correctly extract encoded binary messages from images transfers from a watermarked VFM to its functional copies obtained via pruning and fine tuning; at the same time, we experimentally show that non-watermarked VFMs do not share this property.

Keywords: visual foundation models; neural network watermarking; expressive layers; massive activations; trustworthy artificial intelligence.

For citation: Chistyakova A.S., Pautov M.A. ExpressPrint: An Approach to Watermarking of Visual Foundation Models. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 223-236. DOI: 10.15514/ISPRAS-2025-37(6)-31.

ExpressPrint: метод создания цифровых водяных знаков для визуальных базовых моделей

[1,2] *А.С. Чистякова, ORCID: 0000-0003-4896-4418 <a.chistyakova@ispras.ru>*
[1,3] *М.А. Паутов, ORCID: 0000-0003-0438-6361 <pautov@airi.net>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.*

[2] *Московский государственный университет имени М.В. Ломоносова, Россия, 119991, Москва, Ленинские горы, д. 1.*

[3] *AIRI, Россия, 123112, Москва, Деловой комплекс «Империя», Пресненская набережная, д. 6, стр. 2, 4 этаж.*

Аннотация. Значительные затраты на обучение визуальных базовых моделей с нуля на больших и обширных наборах тренировочных данных мотивируют владельцев моделей прибегать к использованию методов защиты интеллектуальной собственности. В данной работе предложен метод ExpressPrint – новый подход к созданию цифровых водяных знаков для визуальных базовых моделей, основанный на дообучении наиболее выразительных слоев модели совместно с небольшой нейронной сетью типа “кодировщик-декодировщик” с целью встраивания цифровых водяных знаков в отложенный набор входных изображений. Предложенный метод подразумевает незначительные модификации выразительных слоев модели наряду с обучением нейронной сети типа “кодировщик-декодировщик” для извлечения специфичных для пользователя бинарных сообщений из скрытых представлений входных изображений. Данный подход позволяет отличать модель, предоставленную в пользование по лицензии, от других версии модели, и, таким образом, предотвращать несанкционированное использование модели третьими лицами. В работе было обнаружено, что способность корректно извлекать закодированные бинарные сообщения из изображений передается от исходной базовой модели, к ее функциональным копиям, полученным посредством дообучения и прунинга; помимо этого показано, что независимые визуальные базовые модели, не подвергавшиеся нанесению цифровых водяных знаков, не обладают данным свойством.

Ключевые слова: визуальные базовые модели; цифровые водяные знаки для нейронной сети; выразительные слои; массивные активации; доверенный искусственный интеллект.

Для цитирования: Чистякова А. С., Паутов М. А. ExpressPrint: метод создания цифровых водяных знаков для визуальных базовых моделей. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 223–236 (на английском языке). DOI: 10.15514/ISPRAS–2025–37(6)–31.

1. Introduction

Although foundation models are useful tools that are deployed in a variety of practical scenarios from the fields of natural language processing [1], computer vision [2], biology [3], and many others [4], their training is costly in terms of both time and money, making the models valuable assets of their owners. Nowadays, user access to foundation models is mainly organized via subscription to a service where the model is deployed or via purchasing the license to use the specific instance of the model. Unfortunately, some users are violating the terms of use (for example, by integrating their instances of models into their services to make a profit or impermissibly distributing copies of the model). Consequently, it is reasonable that the models’ owners are willing to defend their intellectual property from unauthorized usage by third parties.

One of the prominent approaches to protecting the intellectual property rights (IPRs) of models is watermarking [5-7], a family of methods to embed specific information into the source model by modification of the latter. Ownership verification is then done by checking the existence of this information in the suspicious model. The other set of methods to protect IRPs is based on fingerprinting, which usually does not introduce any changes to the defending model [8-11]. Instead, these methods create a unique feature, or the fingerprint, of the source model; ownership verification

in this case is done by comparing the fingerprints of the defended model and the one extracted from the suspicious model.

In this work, we propose a method to watermark visual foundation models by embedding digital watermarks into hidden representations of certain input images. To choose a proper hidden representation to embed a watermark into, we utilize the concept of massive activations [12]: some blocks of a VFM tend to produce high-magnitude activations in response to various inputs; these activations usually dominate the ones of the subsequent layers. Here and below, we refer to the block (or layer) that produces massive activations to the expressive block (or layer). In our method, we experimentally verify that embedding a watermark into the representation of the expressive block allows us to protect the ownership of VFMs fine-tuned for different practical tasks, such as image classification and segmentation.

Our contributions are summarised as follows:

- We introduce ExpressPrint, a novel approach to watermarking visual foundation models. The proposed method is based on embedding digital watermarks into a hidden representation of a private set of input images of the VFM, where the choice of the proper representation is done by utilizing the concept of massive activations.
- We experimentally show that the proposed method allows us to distinguish between the watermarked VFM and the independent ones under the fine-tuning of the model for different practical tasks, such as classification and segmentation.
- We demonstrate that different VFM architectures can be watermarked by our method, showing the practical applicability of ExpressPrint. This work is the first, to the best of our knowledge, that addresses the problem of watermarking of visual foundation models.

2. Related Work

2.1 Foundation Models and Massive Activations

Visual foundation models, especially those based on Vision Transformers (ViT) [13], have become a dominant paradigm in modern computer vision due to their scalability and transferability across tasks. The development of Self-Supervised Learning (SSL) [14] methods in computer vision has led to the era of universal models, such as SimCLR [15], DINO [16], CLIP [17], and DINOv2 [18], that learn representations from unlabeled images and show impressive flexibility in solving diverse tasks with minimal labeled data for fine-tuning. However, the internal mechanisms, particularly the nature of neural activations, have so far remained little studied.

One emerging concept in the analysis of these models is massive activations [12] – unusually high responses in specific layers or tokens that play a significant role in decision-making. These activations tend to appear across various layers, often have consistently high magnitudes, and are frequently located at the same spatial or token positions across diverse input samples.

2.2 Protecting Intellectual Property via Watermarking and Fingerprinting

The application of watermarking and fingerprinting techniques to protect the intellectual property rights of neural networks has recently become an important topic in trustworthy artificial intelligence. In [19], the authors apply instruction tuning to fingerprint large language models: a predefined private key triggers a model to produce a specific text when present in the input prompt. The authors of [20] formalize the definition of artifact and fingerprint in large generative models based on the geometric properties of the training data manifold. In [8], authors propose to utilize artificially generated images in the attribution of image classifiers under model extraction attacks. In [21], authors discover that it is possible to reliably detect was trained on a synthetic output of a watermarked large language model, which discloses a potential privacy concern of neural network watermarking.

3. Problem Statement

In this work, we present a method to watermark visual foundation models by training an auxiliary network that embeds binary messages into hidden representations of input images of the source VFM. We start by introducing the notations used throughout the paper. Namely, let s be the dimension of an image, Ω be the space of visual foundation models, $f: R^s \to R^d$ be the source VFM that maps input images to embeddings of dimension d, h be the dimension of hidden image representation, and let $m \in \{0,1\}^k$ be the binary vector of length k. Let f be the composition $f(x) \equiv q(p(x))$, where $p: R^s \to R^h$ and $q: R^h \to R^d$ represent mappings from an image to the hidden representation and from the hidden representation to the output embeddings, respectively. In our method, we train two auxiliary models, namely, encoder $e: R^h \times \{0,1\}^k \to R^h$ that embeds the binary message m into the hidden representation $p(x)$ and decoder $d: R^d \to \{0,1\}^k$ that extracts binary messages from $q(x)$. In addition, we fine-tune the latter part of the source model, namely, q. Given the image x, the message m embedded into $p(x)$ and transform $\pi: \Omega \to \Omega$ that maps the foundation model to its functional copy, the goal of the method is two-fold: on the one hand, the decoder d should extract close messages from hidden representations of f and $\pi(f)$; on the other hand, given the model g which is functionally independent of f, the messages extracted from hidden representations of f and g should be far apart.

4. Proposed Method

We introduce ExpressPrint, a novel watermarking method designed to verify ownership of VFMs. ExpressPrint embeds user-specific binary signatures into internal feature representations of VFMs through fine-tuning a small number of expressive layers, accompanied by the joint training of lightweight encoder and decoder networks. This approach enables ownership verification by extracting digital fingerprints directly from model activations when provided with specific input images.

Unlike traditional watermarking techniques that modify model weights or outputs, our method introduces minimal architectural changes while preserving the functional capacity of the model.

4.1 Watermarking Pipeline

The watermarking pipeline is illustrated in Fig. 1. The process consists of the following steps:

1) Embedding of the watermark. Given input image x and user-specific binary message m, we use a lightweight encoder network that injects m into a selected channel of the internal activation of the predefined expressive block. This injection is performed via a forward hook attached to an expressive transformer block.
2) Propagation and Decoding. The modified representation of x is propagated through the remaining trainable layers of the VFM. At a later block, another hook triggers a decoder network to extract the binary message m', which is then compared to m.
3) Training. The encoder, decoder, and a small set of trainable VFM layers are optimized jointly to minimize the decoding loss while maintaining task performance. The remaining VFM weights are frozen to prevent degradation.

4.2 Expressive Layer Selection

The core idea behind ExpressPrint is to embed binary signatures into expressive regions of a model's latent space. We build on the observation that massive activations tend to emerge in later blocks of VFMs and appear for the majority of the input images. Hence, we hypothesize that these high-activation regions are suitable for watermark embedding due to their huge impact on the image representations in the subsequent blocks of the model [12].

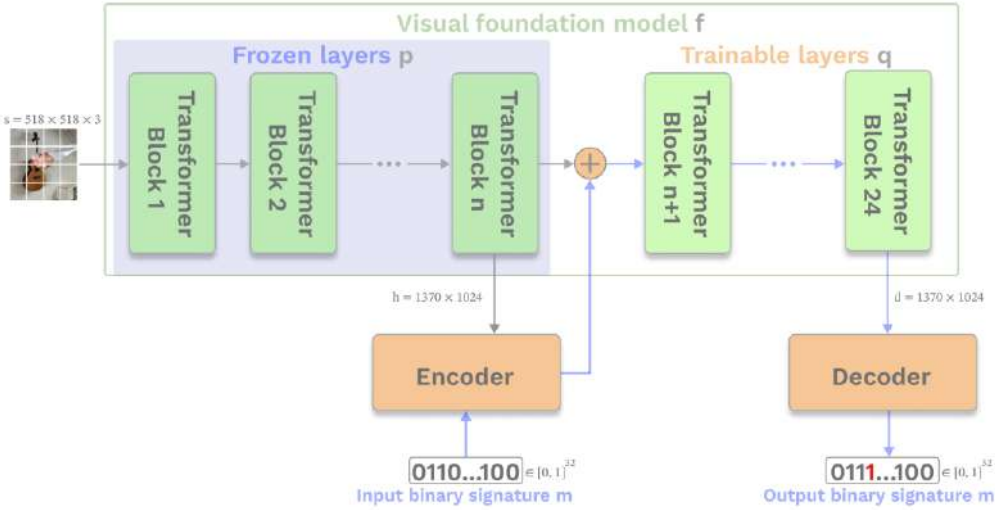


Fig. 1. Schematic illustration of the proposed method. To embed a watermark, we use an auxiliary learnable encoder network and inject a user-specific binary signature into a selected channel of the internal activation. To extract the watermark, we apply an auxiliary decoder network that extracts a binary message from the image representation in a later transformer block.

To identify such regions, we analyze the activation patterns across the blocks of a pre-trained VFM. Specifically, we pass 100 randomly selected natural images through the model and compute, for each block, the average of the top-5 absolute activation values per image. These per-block averages are then aggregated over all images to yield a global activation profile. As shown in Fig. 2, we observe an explosion of activations in the final blocks of the model architecture. This motivates our selection of these blocks as carriers for signature embedding. For each selected expressive block, we further localize the most influential tokens – the ones that are most critical for the block's output. For each token, we compute the absolute values of output activations and choose the token as an outlier if the corresponding output activation increases drastically after some block (namely, if on a particular layer its z-score increases up to 100). We propose to use these outlier tokens as internal activation anchors for binary message injection.

4.3 Loss Function

Our training objective is the combination of two terms: given an input sample, the first one ensures that the feature representations of the watermarked and original models do not deviate much; the second term forces the extracted binary message to be close to the embedded one. Specifically, given $q = q(x)$ as the representation of the expressive part of the source model and $q' = q'(x)$ as the representation of the expressive part of the watermarked model, the objective function is presented in the form below:

$$L(q, q', x, m, m') = L_{fp}(q, q', x) + \lambda L_{sig}(m, m'), \quad (1)$$

where:

- $L_{fp}(q, q', x) = ||q(x) - q'(x)||_2$ is the feature preservation loss (namely, mean squared error between the original representation and the modified representation);
- $L_{sig}(m, m') = ||m - m'||_2$ is the soft distance between the extracted and ground-truth signatures (namely, mean squared error between the extracted and ground-truth signatures);
- λ is a scalar parameter that controls the trade-off between feature fidelity and signature reconstruction.

This formulation ensures that embedded watermarks are extracted from the fingerprinted models while minimizing the impact on the feature representation.

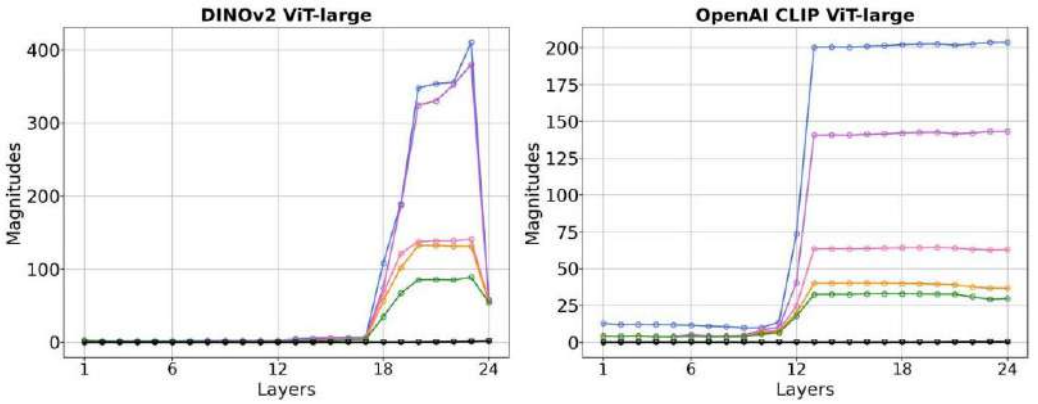


Fig. 2. Top-5 activation values across layers. It is noteworthy that starting from a particular layer, the magnitudes of activations increase drastically. In our work, such a layer represents the beginning of the expressive part of a visual foundation model.

4.4 Assessing the Performance of the Watermarking Method

To evaluate the performance of the proposed method, given a specific user message m and input image x, we compute the distance ρ between the extracted binary message $m'(f,x)$ and m. Note that we specifically indicate that the extracted message depends both on the input image and the model from which it is extracted. We measure the distance as the number of bits which differ in m and $m'(f,x)$:

$$\rho(m, m'(f,x)) \; = \; \sum_{i=1}^{n} \quad 1[m_i(f,x) \neq m'_i], \tag{2}$$

where 1 is the indicator function.

Recall that a good watermarking method has to satisfy two conditions: on the one hand, given an input image x, for the watermarked model f, the distance has to be close to 0; on the other hand, for a separate (independent) model, the distance has to be close to n.

In this work, the decision rule that is used to evaluate whether the given network is watermarked is the comparison of the distance with a predefined threshold: given a single input image x, we treat f as the watermarked iff

$$\rho(m, m'(f,x)) \; <= \; t, \tag{3}$$

where $t >= 0$ is the threshold value. The case of many samples is discussed in Section 5.3.

4.4.1 Setting the Threshold Value

We set the threshold by formulating a hypothesis test: the null hypothesis, $H_0 =$ "$the\ model\ f\ is\ not\ watermarked$" is tested against an alternative hypothesis, $H_1 =$ "$the\ model\ f\ is\ watermarked$", for a given model f. In this work, we assume that the messages $m'(g,x)$ extracted from all the non-watermarked models g are distributed uniformly over all bit strings of length n. Having said so, we estimate the probability of false acceptance of hypothesis H_1 (namely, FPR_1) as follows:

$$FPR_1 = P_{m'(g,x)}(\rho(m, m'(g,x)) \; <= \; t) \; = \; \sum_{q \in [0,\dots,t]} \quad {C_n}^q (1-p)^q p,^{n-q} \tag{4}$$

where $p = P_{m\prime(g,x)}(m_i = m'(g,x)_i)$ is the probability that i'th bits in m and $m'(g,x)$ coincide. In our work, we experimentally evaluate that p is close to 0.5 for all the indices i. To choose a proper threshold value for t, we set up an upper bound for FPR_1 as ε and solve for t, namely,

$$t = arg\ \max_{t'<k}(\sum_{q\in[0,\dots,t']} C_n{}^q(1-p)^q p^{n-q}), subject\ to\ \sum_{q\in[0,\dots,t']} C_n{}^q(1-p)^q p^{n-q} < \varepsilon \quad (5)$$

For example, if $n = 32$ and $\varepsilon = 10^{-5}$, then $t = 4$.

5. Experiments

5.1 Models and Datasets

We conducted our experiments using two large-scale VFMs: CLIP [17] and DINOv2 [18]. For evaluation, we used a subset of images from the ImageNet dataset [22]. To train models on downstream tasks (namely, for classification and segmentation), we utilized three domain-specific datasets:

- E-commerce Product Images [23]: This dataset consists of 18,175 product images categorized into 9 major classes based on Amazon's product taxonomy. It is primarily used for image-based product categorization.
- Oxford-IIIT Pet Dataset [24]: A classification and segmentation dataset containing 37 pet categories (dogs and cats), with approximately 200 images per class. It includes both breed labels and foreground-background segmentation masks.
- FoodSeg103 [25]: A food image segmentation dataset containing 7,118 images annotated with fine-grained pixel-wise labels for over 100 food categories. It supports both semantic segmentation and instance-level analysis of food items.

5.2 Training Details

5.2.1 Watermark Injection

We initialized both VFMs with publicly available pre-trained weights [27-28]. To embed and extract binary watermarks within their internal representations, we designed a lightweight encoder-decoder architecture. For each image, we randomly and uniformly sampled a 32-bit binary vector and assigned it as the corresponding watermark for this image.

The watermark encoder consisted of two fully connected layers and processed the concatenation of a selected feature channel with the binary watermark vector. Namely, the encoder maps the pair (internal image representation, binary message) to a vector from the dimension of the hidden representation:

$$e(p(x), m) = u \in R^h. \quad (6)$$

This encoded perturbation was injected into a specific transformer block and channel via a forward hook.

Similarly, the watermark decoder, also composed of two fully connected layers, was used to reconstruct the watermark from the modified features extracted at a later block. During inference, outputs were thresholded at 0.5 to obtain binary predictions:

$$d(q(e(p(x), m)))) = v \in R^k;\ m'_i = 1(v_i > 0.5)\ \forall\ i \in [1, \dots, k]. \quad (7)$$

Based on prior activation analysis (see Fig. 2), we chose block 12 in CLIP ViT-L/14 and block 18 in DINOv2 ViT-L/14 for the embedding of the watermarks (note that in the corresponding blocks, massive activations first emerge). All transformer layers prior to the embedding point were frozen during training, while the encoder, decoder, and all subsequent blocks are subject to change during watermark embedding. Further discussion of this selection is provided in Section 5.3.

During training, we used AdamW optimizer with learning rates of 10^{-4} for the VFM backbones and 10^{-5} for watermark modules (namely, e and d). Both models were trained for 5 epochs with a batch size of 16.

5.2.2 Fine-tuning for Downstream Tasks

We used a full fine-tuning strategy in all experiments to preserve watermark robustness against representation shifts introduced by fine-tuning. Experiments were conducted on both classification and segmentation tasks. For the classification tasks, three learning rate schedulers were evaluated: constant (no scheduler), cosine annealing, and linear decay; for the segmentation tasks, no learning rate schedulers were used. The training was performed using the AdamW optimizer for 10 epochs.

5.2.3 Pruning

To investigate the impact of model sparsity on both classification accuracy and watermark robustness, we applied post-training unstructured L1-norm pruning to the entire model. We evaluated two sparsity levels: moderate pruning, where 20% of the lowest-magnitude weights were zeroed out, and aggressive pruning, where 40% of the weights were removed. This procedure enabled us to assess the effect of varying sparsity levels on watermark reconstruction. Note that the unrestricted L1-norm pruning is used purely as the baseline to illustrate the robustness of the proposed method to a model's modification.

5.3 Signature Location Selection

Given the source model f, the set of input samples $\{x_1, x_2, \dots, x_N\}$, false positive rate threshold from Eq. (4). and corresponding distance threshold t from Eq. (5), we compute the watermark detection rate

$$r = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[\rho(m(x_i), m'(f, x_i)) <= t], \quad (8)$$

where $N = 1000$. Note that here we explicitly write $m(x_i)$ to indicate that for different input images, watermark messages may differ.

For the length of the watermark $k = 32$ and two similarity thresholds (namely, $t = 0$ and $t = 4$), we study the dependence of the watermark detection rate on the number n of the last frozen layer (see Fig. 1 for clearance). Note that the value of the watermark detection rate at $t = 0$ indicates the ability of the method to extract the same watermarks that were embedded. To evaluate the robustness of the embedded watermarks to downstream task adaptation, we fine-tuned the CLIP model on a semantic segmentation task (see Section 5.2 for details). The values of the hyperparameters were chosen to satisfy the trade-off between the feasibility of the watermark embedding and its robustness under the perturbations of the watermarked model.

In Table 1, we report the watermark detection rates and average bitwise distance between an embedded watermark and an extracted binary message (see Eq. (2)).

The results indicate that early transformer blocks have low watermark detection rates and high reconstruction errors, making them unsuitable for embedding. In contrast, blocks 12, 13, 15, 21, and 22 show high detection rates and low errors, demonstrating better stability and reliability for watermark embedding. Block 12, in particular, provides the best balance between detection accuracy and reconstruction quality. Notably, this is also the first layer where massive activations begin to emerge, which may contribute to its effectiveness as an embedding point.

While watermarks embedded closer to the end of the transformer (e.g., in blocks 21 and 22) are extracted with reasonably high accuracy, their robustness degrades after fine-tuning. This is likely because the later layers are more heavily modified during task-specific adaptation, which negatively impacts the detection rate. Therefore, mid-level blocks such as block 12 offer a more reliable trade-off between initial detection performance and robustness to downstream fine-tuning.

Table 1. Dependency of the watermark detection rate on the number of the first expressive transformer block, n. The architecture of the source visual foundation model is CLIP.

Block number, n	Watermark detection rate, $r \uparrow$ $t = 0$	Average bitwise error $\downarrow$	Watermark detection rate after fine-tuning, $r \uparrow$ $t = 4$
1	0.000	15.959	0.000
2	0.000	16.033	0.000
10	0.000	14.382	0.000
11	0.000	11.255	0.000
12	0.938	**0.143**	**0.983**
13	**0.960**	0.607	0.980
14	0.483	1.766	0.610
15	0.940	0.885	0.810
21	0.939	0.328	0.945
22	0.931	0.150	0.864
23	0.569	0.852	0.882
24	0.000	15.905	0.000

6. Results

In this section, we provide the results of experiments and elaborate on them.

6.1 Overall Results

In Fig. 3, we report the dependency of the watermark detection rate on the threshold value of the false positive rate from Eq. (4). We evaluated classification and segmentation tasks and studied the effect of unstructured pruning. Details of experiments are provided in Section 5.2.

6.2 Comparison with Fingerprinting Methods

To assess the effectiveness of the proposed method, we compare it against the state-of-the-art fingerprinting method, ADV-TRA [26]. The main idea of this method goes as follows. Given the input sample x of the ground truth class y, an index of the target class y_t, and the source model f, the set of adversarial examples $T = (x, x_1, x_2, \dots, x_l)$ is computed such that

$$x_{i+1} = x_i - s_i \, sign(\nabla L(f, x_{i,}, y_t)),$$

where L is the loss function guiding the optimization towards y_t, and s_i is the scalar variable denoting the step size on i'th iteration. Then, this set T is used to compare the predictions of the source model f and a suspect model, f_{sus}, to decide whether f_{sus} is a stolen copy of f. It is worth mentioning that the process of generation of the model's fingerprint, namely, the predictions on the adversarial examples, is downstream task-dependent: the owner of the model has to use an auxiliary classification head to compute adversarial examples. This limitation makes ADV-TRA barely feasible for other downstream tasks.

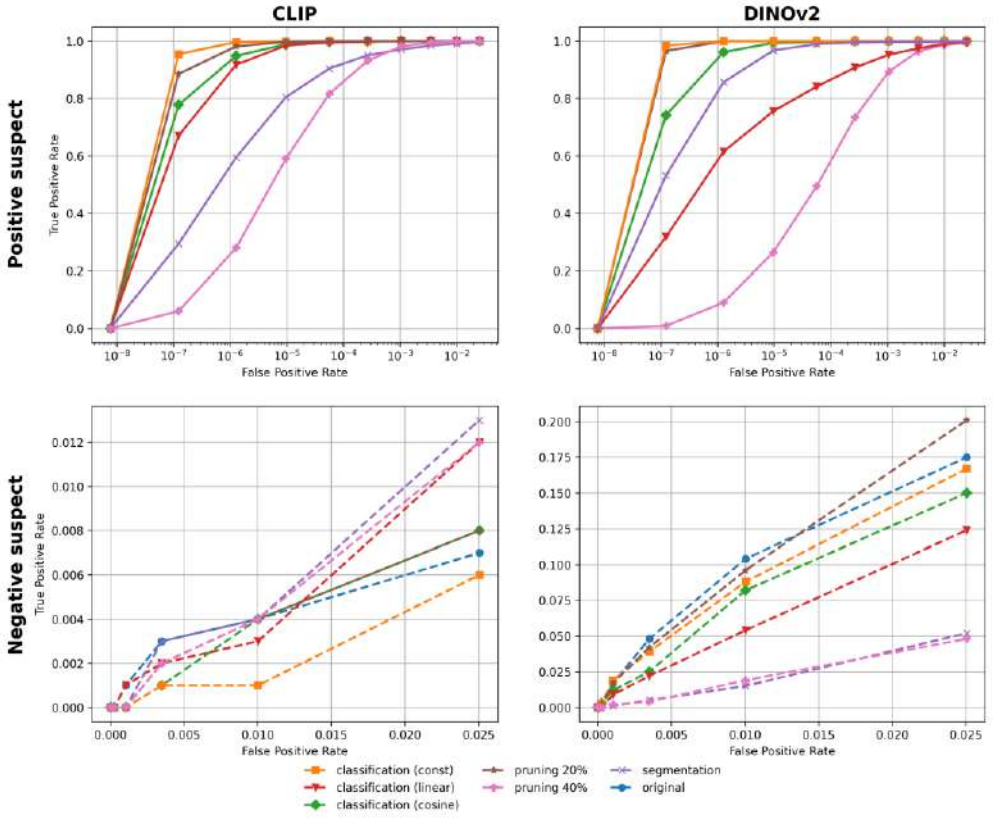


Fig. 3. Effectiveness of ExpressPrint in ownership verification problem. We consider several downstream tasks (classification and segmentation), and VFM architectures (CLIP and DINOv2) and evaluate our approach against fine-tuning and pruning of the watermarked model.

To compare ExpressPrint with ADV-TRA on different downstream tasks and assess their robustness under different types of model perturbations, we compute the watermark detection rate (or fingerprint detection rate, in the case of ADV-TRA). Note that the higher the detection rate for positive suspect models, the more precisely a method detects functionally stolen models; on the other hand, the lower the detection rate in the case of the negative suspect models, the less frequently a method detects an independent model as the functional copy of the source one. In Table 2, we report the quantitative results of the comparison.

There are two types of suspect models in Table 2: positive suspect models and negative suspect models. While positive suspect models represent the set of VFMs that are functionally connected to the source (watermarked) model, the negative suspect models are independent of the source VFM. The description of positive suspect models is presented in Sections 5.2.2. and 5.2.3; as the negative suspect models, we use different visual foundation models, namely, CLIP [17] and DINOv2 with

registers [29]. Note that the difference between DINOv2 and DINOv2 with registers is that the latter is equipped with additional learnable tokens which are shared among different input samples during the model's training.

*Table 2. Comparison of different fingerprinting techniques, the architecture of the source visual foundation model is DINOv2. We report a watermark (fingerprint) detection rate for both positive suspect models and negative suspect models. The best results are highlighted in **bold**. It is noteworthy that ExpressPrint outperforms ADV-TRA both in terms of true positive detection rate and false positive detection rate.*

Model Type		ADV-TRA	ExpressPrint
Positive suspect	classification (const)	0.703	**1.000**
	classification (linear)	0.012	**0.842**
	classification (cosine)	0.123	**0.998**
	segmentation	0.000	**0.990**
	pruning 20%	**1.000**	**1.000**
	pruning 40%	0.086	**0.495**
Negative Suspect	different architecture (DINOv2 with registers)	0.012	**0.000**
	different architecture (CLIP)	**0.000**	**0.000**

In Table 3, we report the computation overhead of the methods. It is worth mentioning that ADV-TRA is excessively time-consuming in terms of fingerprint generation. At the same time, signature verification times (or watermark extraction times for ExpressPrint) do not differ much for these two approaches. Note that the first two columns of the Table 3 (namely, Signature injection (min) and Signature generation (min)) indicate the time required for watermark (or fingerprint) preparation; although for a given model this procedure has to be conducted only once, the faster it is, the more applicable the corresponding method for the large-scale VFMs.

7. Discussion and Limitations

We experimentally show that ExpressPrint allows us to distinguish between independent models and functional copies of the given watermarked visual foundation model. It is worth mentioning that ExpressPrint is a downstream task-agnostic approach: the model's owner has to prepare a set of input images and perform the watermark embedding procedure only once for a given instance of the model; then, the watermarked model remains detectable by our method after fine-tuning to a particular downstream task (for example, image classification and segmentation). In comparison to ExpressPrint, ADV-TRA, the state-of-the-art fingerprinting approach does not provide a satisfactory level of fingerprint detection rate when the fingerprinted model is fine-tuned for a different downstream task (namely, segmentation; see Table 2). More than that, the proposed method is robust to the pruning of the protected model and yields very low false detection rates (namely, zero for the experiments from Table 2).

Table 3. Comparison of computational overheads for ADV-TRA and ExpressPrint, the architecture of the source visual foundation model is DINOv2. We report cumulative time in minutes for N=1000 images.

	Signature injection (min)	Signature generation (min)	Signature verification (min)
ADV-TRA	0.000	1663.70	3.412
ExpressPrint	31.533	3.105	4.017

Among the limitations of ExpressPrint, we indicate the necessity to have white-box access to the first expressive layer of a suspect model: to verify that a model allows to encrypt and decrypt a certain message, one has to pass the image and corresponding binary string to it (see Eq. (7)).

8. Conclusion and Future Work

In this work, we propose ExpressPrint, a novel watermarking approach for visual foundation models. This method is model agnostic and allows to protect a VFM in a downstream task-independent manner: our experiments show that the proposed method allows to reliably detect the functional copies of a particular foundation model obtained by the fine-tuning and pruning both for image classification and segmentation. On the other hand, we verify that ExpressPrint does not detect benign, independent models as functional copies of the watermarked VFM, which makes the method applicable in practical scenarios. We compared the effectiveness and computation overhead of the proposed method with the state-of-the-art fingerprinting approach, ADV-TRA, and showed that the watermark detection rate and false positive detection rate of ExpressPrint are superior to the ones of ADV-TRA; at the same time, the proposed approach is significantly faster. Important directions of future work include an adaptation of the ExpressPrint to allow black-box inference of the suspect models.

References

[1]. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.

[2]. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125, 1(2), 3.

[3]. Ma, Jun, and Bo Wang. "Towards foundation models of biological image segmentation." Nature Methods 20.7 (2023): 953-955.

[4]. Schneider, Johannes, Christian Meske, and Pauline Kuss. "Foundation models: a new paradigm for artificial intelligence." Business & Information Systems Engineering 66.2 (2024): 221-231.

[5]. Uchida, Y., Nagai, Y., Sakazawa, S., & Satoh, S. I. (2017, June). Embedding watermarks into deep neural networks. In Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval (pp. 269-277).

[6]. Guo, J., & Potkonjak, M. (2018, November). Watermarking deep neural networks for embedded systems. In 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD) (pp. 1-8). IEEE.

[7]. Li, Y., Zhu, L., Jia, X., Jiang, Y., Xia, S. T., & Cao, X. (2022, June). Defending against model stealing via verifying embedded external features. In Proceedings of the AAAI conference on artificial intelligence (Vol. 36, No. 2, pp. 1464-1472).

[8]. Pautov, M., Bogdanov, N., Pyatkin, S., Rogov, O., & Oseledets, I. (2024, August). Probabilistically robust watermarking of neural networks. In Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (pp. 4778-4787).

[9]. Quan, Y., Teng, H., Xu, R., Huang, J., & Ji, H. (2023). Fingerprinting deep image restoration models. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 13285-13295).

[10]. He, Z., Zhang, T., & Lee, R. (2019). Sensitive-sample fingerprinting of deep neural networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 4729-4737).
[11]. Peng, Z., Li, S., Chen, G., Zhang, C., Zhu, H., & Xue, M. (2022). Fingerprinting deep neural networks globally via universal adversarial perturbations. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 13430-13439).
[12]. Sun M., Chen X., Kolter J. Z., and Liu Z. Massive Activations in Large Language Models. (2024) In First Conference on Language Modeling.
[13]. Dosovitskiy A., Beyer L., Kolesnikov A., Weissenborn D., Zhai X., Unterthiner T., Dehghani M., Minderer M., Heigold G., and Gelly S. (2020) An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In International Conference on Learning Representations, 2020.
[14]. Balestriero R., Ibrahim M., Sobal V., Morcos A., Shekhar S., Goldstein T., Bordes F., Bardes A., Mialon G., Tian Y., Schwarzschild A., Wilson A. G., Geiping J., Garrido Q., Fernandez P., Bar A., Pirsiavash H., LeCun Y., and Goldblum M. (2023) A Cookbook of Self-Supervised Learning. arXiv:2304.12210.
[15]. Chen T., Kornblith S., Norouzi M., and Hinton G. (2020) A simple framework for contrastive learning of visual representations. In International conference on machine learning, PmLR, 2020, pp. 1597–1607.
[16]. Caron M., Touvron H., Misra I., Jégou H., Mairal J., Bojanowski P., and Joulin A. (2021) Emerging properties in self-supervised vision transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9650–9660.
[17]. Radford A., Kim J. W., Hallacy C., Ramesh A., Goh G., Agarwal S., Sastry G., Askell A., Mishkin P., and Clark J. (2021) Learning transferable visual models from natural language supervision. In International conference on machine learning, PmLR, 2021, pp. 8748–8763.
[18]. Oquab M., Darcet T., Moutakanni T., Vo H., Szafraniec M., Khalidov V., Fernandez P., Haziza D., Massa F., and El-Nouby A. (2024) DINOv2: Learning Robust Visual Features without Supervision, Trans. Mach. Learn. Res. J., 2024, pp. 1–31.
[19]. Xu, J., Wang, F., Ma, M., Koh, P. W., Xiao, C., & Chen, M. (2024, June). Instructional Fingerprinting of Large Language Models. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers) (pp.-3277-3306).
[20]. Song, H. J., Khayatkhoei, M., & AbdAlmageed, W. (2024). Manifpt: Defining and analyzing fingerprints of generative models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10791-10801).
[21]. Sander, T., Fernandez, P., Durmus, A., Douze, M., & Furon, T. (2024). Watermarking makes language models radioactive. Advances in Neural Information Processing Systems, 37, 21079-21113.
[22]. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp.-248-255).
[23]. Ecommerce Product Images for Product Categorization - https://www.kaggle.com/datasets/fatihkgg/ecommerce-product-images-18k, accessed 13.05.2025.
[24]. The Oxford-IIIT Pet Dataset - https://www.kaggle.com/datasets/tanlikesmath/the-oxfordiiit-pet-dataset, accessed 13.05.2025.
[25]. FoodSeg103 - https://huggingface.co/datasets/EduardoPacheco/FoodSeg103, accessed 13.05.2025.
[26]. Xu, T., Wang, C., Liu, G., Yang, Y., Peng, K., & Liu, W. (2024). United We Stand, Divided We Fall: Fingerprinting Deep Neural Networks via Adversarial Trajectories. Advances in Neural Information Processing Systems, 37, 69299-69328.
[27]. CLIP model for timm - https://huggingface.co/timm/vit_large_patch14_clip_224.openai, accessed 16.05.2025.
[28]. DINOv2 model for timm - https://huggingface.co/timm/vit_large_patch14_dinov2.lvd142m, accessed 16.05.2025.
[29]. Darcet, T., Oquab, M., Mairal, J., & Bojanowski, P. Vision Transformers Need Registers. In The Twelfth International Conference on Learning Representations.

Информация об авторах / Information about authors

Анна Сергеевна ЧИСТЯКОВА – старший лаборант Центра доверенного искусственного интеллекта ИСП РАН. Научные интересы: исследование и разработка нейросетевых архитектур, устойчивых к состязательным атакам.

Anna Sergeevna CHISTYAKOVA – Senior Assistant at ISP RAS Research Center for Trusted Artificial Intelligence. Research interests: research and development of neural network architectures aimed at robustness to adversarial attacks.

Михаил Александрович ПАУТОВ – кандидат компьютерных наук, научный сотрудник AIRI, Центра доверенного искусственного интеллекта ИСП РАН. Сфера научных интересов: линейная алгебра, теория больших отклонений, доказуемая устойчивость нейронных сетей, цифровые водяные знаки.

Mikhail Aleksandrovich PAUTOV – Cand. Sci. (Phys.-Math.) in computer science, research scientist at AIRI and ISP RAS Research Center for Trusted Artificial Intelligence. His research interests include linear algebra, large deviations theory, certified robustness and digital watermarking.

DOI: 10.15514/ISPRAS-2025-37(6)-32

Shazam Algorithm for Partial Video Copy Detection

[1,2] *R.S. Uzdenov, ORCID: 0009-0006-5221-4950 <498rustam@gmail.com>*
[1] *A.I. Perminov, ORCID: 0000-0001-8047-0114 <perminov@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*

[2] *Bauman Moscow State Technical University, 2-ya Baumanskaya st., 5/1, Moscow, 105005, Russia.*

Abstract. The Shazam algorithm has proven its reliability and efficiency in audio identification tasks. In this paper, we adapt the core principles of the Shazam algorithm for the problem of partial video copy detection. We propose a novel method for alignment video fingerprints in partial copy detection search of video query across video base. One of the best features of this method: fast CPU execution, simplicity and at the same time high efficiency. Experimental results on publicly available video datasets demonstrate that our approach achieves high accuracy in detecting partial and modified video copies, with competitive performance in terms of speed and scalability. Our findings suggest that Shazam-inspired fingerprinting can serve as an effective tool for large-scale video copy detection applications.

Keywords: partial video copy detection; shazam algorithm; perceptual hashing; keyframe extraction; video fingerprinting; nearest neighbor search; media forensics; copyright protection; large-scale video analysis; realtime detection; cpu-efficient algorithms; open-source video search.

For citation: Uzdenov R.S., Perminov A.I. Shazam Algorithm for Partial Video Copy Detection. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 6, part 2, 2025, pp. 237-248. DOI: 10.15514/ISPRAS-2025-37(6)-32.

Алгоритм Shazam для обнаружения частичного видео копирования

[1,2] *Р.Ш.Узденов, ORCID: 0009-0006-5221-4950 <498rustam@gmail.com*
[1] *А.И.Перминов, ORCID: 0000-0001-8047-0114 <perminov@ispras.ru>*
[1] *Институт системного программирования им. В.П. Иванникова РАН,*
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.
[2] *Московский государственный технический университет им. Н. Э. Баумана,*
Россия, 105005, Москва, ул. 2-я Бауманская, 5.

Аннотация. Алгоритм Shazam доказал свою надежность и эффективность в задачах идентификации аудио. В данной работе мы адаптируем основные принципы алгоритма Shazam для задачи обнаружения частичных видеокопий. Мы предлагаем новый метод выравнивания видеоотпечатков при поиске частичной видеокопии запроса по базе видео. Одно из лучших качеств данного метода – его высокая скорость исполнения на CPU, простота и одновременно с этим высокая эффективность. Экспериментальные результаты на общедоступных видео наборах данных демонстрируют, что наш подход достигает высокой точности в обнаружении частичных и модифицированных видеокопий, обладая конкурентной производительностью по скорости и масштабируемости. Наши результаты показывают, что создание отпечатков по принципам Shazam может служить эффективным инструментом для крупномасштабных приложений по обнаружению видеокопий.

Ключевые слова: поиск копий видеофрагментов; алгоритм Shazam; перцептивное хеширование; извлечение ключевых кадров; видео фингерпринтинг; поиск ближайших соседей; медиакриминалистика; защита авторских прав; анализ видео на больших данных; обнаружение в реальном времени; ресурсоэффективные алгоритмы; открытые системы поиска видео.

Для цитирования: Узденов Р.Ш., Перминов А.И. Shazam алгоритм для обнаружения частичного видео копирования. Труды ИСП РАН, том 37, вып. 6, часть 2, 2025 г., стр. 237–248 (на английском языке). DOI: 10.15514/ISPRAS–2025–37(6)–32

1. Introduction

The rapid growth of online video content has made partial video copy detection essential for copyright enforcement, content moderation, and large-scale media search. Real-world copies are often transformed – cropped, re-encoded, overlaid, or altered in brightness, color, or speed – making robust detection challenging.

While many detection methods exist, most are either proprietary, lack open-source transparency, or depend on resource-intensive deep neural networks unsuitable for scalable, real-time processing on standard hardware. Few achieve an optimal balance of robustness, efficiency, and simplicity.

We address this gap by adapting the core principles of the Shazam algorithm [1] from audio to video. Our method extracts keyframes at regular intervals, hashes them using perceptual image hashing, and matches them via efficient approximate nearest neighbor search. Post-processing aligns candidate fragments by analyzing time differences between matches. The approach is fully parallelizable for modern CPUs (Central Processing Unit).

Evaluation on the VCDB (Video Copy Detection Benchmark) benchmark [2] demonstrates that our solution combines accuracy, efficiency, and practical scalability under diverse video transformations

The main contributions of this paper are as follows:

- We present an efficient, open-source implementation of a Shazam-inspired algorithm for partial video copy detection.
- Our pipeline achieves robust detection of copied fragments under a wide range of real-world video modifications, combining simplicity, high speed, and scalability.
- We provide comprehensive experimental results on the VCDB dataset [2], using standard metrics such as Mean Reciprocal Rank (MRR), Mean Average Precision (MAP), and recall.

2. Related Work

Video copy detection and partial video copy detection have been active research areas for more than a decade. The main approaches can be grouped into three broad categories: perceptual hashing, block-based and spatio-temporal signatures, and neural network-based methods. In addition, there are several industrial and open-source systems with varying levels of accessibility and transparency.

2.1 Surveys and Overviews

A number of survey papers have systematically reviewed the landscape of video fingerprinting and copy detection methods. Recent works, such as the 2024 survey "Digital Fingerprinting on Multimedia" (200+ references) [3], provide a taxonomy of fingerprinting techniques, covering classical block-based methods, perceptual hashes, and learning-based approaches. They highlight key requirements for real-world systems: robustness to transformations, computational efficiency, and scalability. Other recent reviews, e.g., "The 2023 Video Similarity Dataset and Challenge," [4] focus on benchmarking and the increasing role of deep learning, including transformer architectures [5], for video retrieval and similarity matching

2.1.1 Block-Based and Spatio-Temporal Signatures

Classical video fingerprinting techniques include methods based on spatial and spatio-temporal signatures. The 2009 Vobile paper [6], for example, uses spatial signatures derived from luminance (Y channel in YUV (Y component (luma) and two chroma components U and V)) and organizes video into blocks, enabling geometric transformation robustness and efficient matching. Robust ordinal measure methods [7] (2004) and TIRI (Temporally Informative Representative Images, 2009) [8] extract features from carefully sampled frames or generate 3D hashes to capture temporal context. However, such methods are often designed for full-video or coarse fragment detection, may not be robust to all types of edits, and sometimes lack public, well-maintained code.

2.1.2 Perceptual Hashing and Open-Source Libraries

Perceptual image hashing methods are popular for their simplicity, speed, and small storage footprint. Tools like OpenCV [9] and libraries such as ImageHash [10] offer a range of hash algorithms (pHash, dHash, whash, etc.). These are widely used for frame-level fingerprinting in video search pipelines. Benchmarks indicate that while CNN-based hashes [11] can provide higher accuracy, classic hashes remain competitive for many tasks. However, none of the basic image hash functions is fully robust to geometric transformations (e.g., flipping, severe cropping).

Recent open-source projects, such as VideoHash [12] and ViDeDup [13], provide practical implementations for near-duplicate detection, mainly focusing on the whole video or image collections, but often lack support for fine-grained fragment matching and may not be robust against severe modifications.

2.1.3 Neural and Learning-Based Methods

The latest advances in partial copy detection use neural networks to learn robust video representations. Winners of the DVSC23 (Dataset of Video Similarity Challenge 2023) [4] challenges have used large transformer models and deep similarity networks, achieving top performance in complex, large-scale retrieval settings. While these approaches achieve strong robustness, they require significant computational resources and are not always feasible for lightweight, real-time scenarios.

2.2 Proprietary and Industrial Systems

Many effective commercial systems exist, including those by Vobile [6] and Microsoft's PhotoDNA [14]. These are typically proprietary and designed for industrial applications (copyright, anti-piracy,

CSAM (Child Sexual Abuse Material) (detection). While they set a high standard in robustness and deployment scale, the lack of transparency and public implementation limits their use in academic or open-source projects.

2.2.1 Shazam Algorithm and Audio Fragment Alignment

The Shazam algorithm [1], though originally designed for audio fingerprinting, is notable for its speed, fragment-level matching, and alignment capabilities. Its open principles have inspired similar approaches in video, including the method presented in this paper. Audio fingerprint alignment is a key idea, enabling not just duplicate detection but robust localization of copied fragments.

2.3 Benchmarks and Datasets

Several public datasets and benchmarks support research in this area. The VCDB dataset [2] is specifically designed for partial copy detection and is widely used for evaluation, have 27 hours of video content from YouTube and MetaCafe with 9k+ pairs of similar segments. DVSC23 [4] is a newer, more complex benchmark but may exceed the scale needed for lightweight systems. Proper evaluation relies on standard metrics such as Mean Reciprocal Rank (MRR), Mean Average Precision (MAP), recall, and found original accuracy.

2.4 Summary of Practical Implications

Open-source solutions do not yet achieve a balance of simplicity, CPU efficiency, robustness to video modifications, and fragment-level search. Neural approaches deliver strong accuracy but require significant resources. Most practical solutions use image/frame hashing, with search and alignment inspired by Shazam-like approaches (see Table 1).

Table 1. Summary table of different methods/tools.

Method/Tool	Approach	Key Features / Limitations	Open Source
VideoHash [12]	64-bit video hash	Fast, low memory, no fragment support, not robust to flips	Yes
VideoDeduplication [13]	DCT + clustering	Old code, not maintained, focus on full video	Yes
Wechat CV VSC2022 [15]	Deep learning (transformer)	SOTA accuracy, high complexity, high resource requirements	Yes
pyPhotoDNA [14]	Microsoft algorithm	Proprietary, large vector, slow, robust to attacks	Partially
Vobile (industry) [6]	Block-based signature	Proprietary, robust, industrial standard	No

3. Problem Statement

The proliferation of digital video content has made the ability to automatically identify copied or reused video fragments across vast multimedia collections a problem of global importance. Partial video copy detection is a foundational technology for protecting intellectual property, supporting copyright enforcement, enabling digital rights management, and combating misinformation and illicit content distribution. As video sharing and remix culture become integral to communication, a robust and efficient solution is essential for ensuring fair use and trust in digital media.

3.1 Task Definition

Given a query video fragment Q and a large database of reference videos $\mathcal{D} = \{V_1, V_2, \ldots, V_N\}$, the goal is to:

- **Detect** all segments in $\mathcal{D}$ that contain a fragment visually similar to Q, even if the fragment has been altered (cropped, re-encoded, overlaid, color modified, etc.).
- **Localize** the temporal boundaries of each detected copy, i.e., determine the start and end times within the reference video where the copy occurs.

3.2 Formal Statement

Let Q be a query video fragment of arbitrary length, and V_i a video from the reference database. The system must identify all tuples $(V_i, t_{start}, t_{end})$such that the segment $V_i[t_{start}: t_{end}]$ is a copy of Q (or a substantially similar transformation), subject to some robustness threshold.

3.3 Challenges

- **Transformations:** Real-world copies may undergo various visual and temporal modifications (cropping, color jitter, overlays, speed changes, compression, geometric transforms, etc.).
- **Scalability:** The reference database may contain millions of videos, requiring solutions that are both memory- and CPU-efficient.
- **Fragment Alignment:** It is insufficient to detect only the presence of a copy; precise temporal alignment is needed to localize the copied fragment.

3.4 Evaluation Metrics

Performance is typically measured using:

- **Mean Reciprocal Rank (MRR):** Measures how quickly a relevant video is retrieved.
- **Mean Average Precision (mAP):** Captures overall ranking quality.
- **Recall:** Fraction of true copies correctly detected.
- **Found Original:** Whether the exact original is retrieved and localized.

A practical solution must achieve high accuracy across these metrics under diverse transformation conditions, while maintaining computational efficiency suitable for real-time or large-scale applications.

4. Proposed Method

This section details the proposed approach for partial video copy detection, inspired by the alignment and fingerprinting techniques of the Shazam audio algorithm [1]. The method is designed to be computationally efficient, scalable to large video collections, and robust to typical video transformations. The pipeline consists of four key stages:

1. **Keyframe Extraction**: Systematic sampling of representative frames from video sequences.
2. **Fingerprinting and Hashing**: Generation of compact, perceptual descriptors for each keyframe.
3. **Search and Alignment**: Fast identification and temporal alignment of matching fragments within a reference database.

4.1 Keyframe Extraction

A fundamental component of the proposed method is the systematic extraction of representative keyframes from each video. Keyframe extraction serves two principal goals: reducing data redundancy and enabling robust comparison between videos based on compact, meaningful content descriptors.

For both reference and query videos, frames are sampled at regular temporal intervals, typically every 0.5 seconds. This uniform sampling strategy provides a balance between computational efficiency and the ability to capture temporal variations, even in videos with scene changes or edits. The use of regular intervals, rather than scene-change detection, ensures that the approach is simple, reproducible, and does not depend on the availability or accuracy of more complex scene detection algorithms. This procedure is illustrated in Fig. 1 (first cycle), which presents the flowchart for keyframe extraction and preprocessing. Keyframes extracted at this stage are used for both constructing the reference fingerprint database and for processing query video fragments.

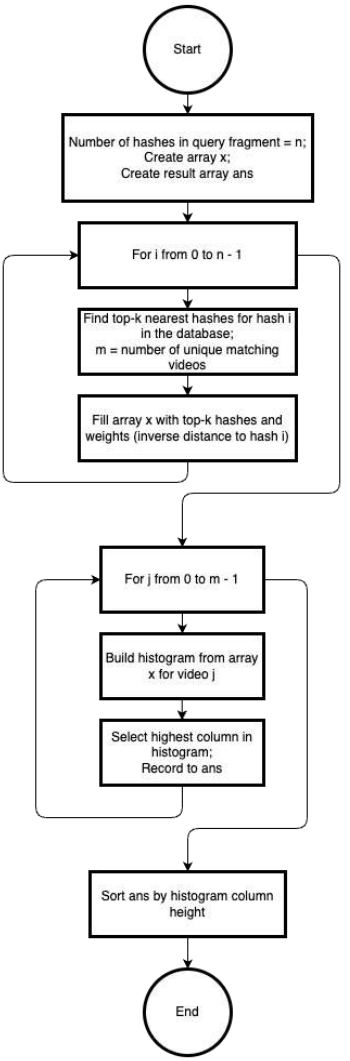


Fig. 1. Pipeline of indexing and search video.

4.2 Fingerprinting and Hashing

Following keyframe extraction, each keyframe is transformed into a compact and robust fingerprint using perceptual image hashing algorithms. The goal of this stage is to generate a representation that is both discriminative-enabling distinction between different content– and robust to common video transformations such as compression, resizing, or moderate changes in color and brightness.

For every extracted keyframe, a hash function (e.g., pHash, dHash, or other perceptual hash) is applied to produce a fixed-length descriptor. The choice of hash function is guided by the need for:

- **Robustness** to minor and color image modifications;
- **Compactness** for efficient storage and fast search;
- **Simplicity** for reproducible implementation on standard hardware.

Each hash value is associated with the corresponding video identifier and the timestamp of the keyframe. In the context of reference videos, these triples (video ID, timestamp, hash) are stored in the fingerprint database. For a query video, the resulting sequence of hashes is used as the search template.

This hashing procedure enables scalable similarity search by mapping keyframes into a common feature space where visually similar frames are close under a chosen distance metric (e.g., Hamming or Euclidean distance). The use of perceptual hashing allows the system to tolerate small transformations and degradations, which are frequently encountered in user-generated or re-encoded video content.

This stage is illustrated in the middle part of Fig. 1, where each keyframe is processed by the hash function, and the resulting fingerprint is recorded for subsequent matching and alignment.

4.3 Search and Alignment

The search and alignment stage is responsible for identifying and temporally localizing potential video copies within a large reference database. This is achieved through efficient similarity search and a robust alignment procedure inspired by the Shazam algorithm for audio [1].

4.3.1 Similarity Search

For each hash in the query sequence, the system performs a nearest neighbor search in the fingerprint database, typically retrieving the top-k closest matches based on a chosen distance metric (e.g., Hamming or Euclidean distance). Efficient indexing structures, such as Annoy, are used to enable sublinear search time even for millions of fingerprints. Each database match provides not only a candidate video ID but also the timestamp of the matched keyframe.

4.3.2 Temporal Alignment

To identify true fragment-level copies and distinguish them from incidental matches, the method aggregates all retrieved matches by calculating the time offset between the position of the query frame and the corresponding database frame. For each candidate reference video, a histogram of these offsets is constructed. Peaks in the histogram represent time shifts where multiple query frames align consistently with the same region of a reference video– strong evidence of a copied segment.

4.3.3 Fragment Localization and Ranking

The location of the highest peak in the offset histogram determines the estimated start time of the copied segment in the reference video. The height of the peak (the weighted sum of matching frames at the same offset) is used as a confidence score. Candidate videos are ranked according to this score, and the corresponding time intervals are reported as the detected copy locations.

This approach is robust to missing or noisy matches, as the alignment mechanism accumulates evidence over multiple frames. It further enables partial and modified copies to be detected and localized with high precision, even in the presence of typical video transformations.

The overall search and alignment process is visualized in Fig. 1, illustrating the construction of the offset histogram and the identification of the optimal alignment between the query and reference video sequences.

5. Experimental Setup

Given the computational intensity of keyframe extraction, hashing, and nearest neighbor search, we employ aggressive parallelization to maximize throughput and minimize wall-clock time.

5.1 Dataset

We evaluated our method on the VCDB dataset [2], which contains annotated pairs of partially overlapping real, not simulated video fragments subjected to various transformations such as cropping, overlays, changes in brightness, color jitter, rescaling, and re-encoding.

5.2 Frame Hash Functions and Search Algorithm

The search algorithm follows a “Shazam-style” pipeline:

1. Extract and hash keyframes for all database and query videos
2. Index all database hashes using Annoy for fast ANN search.
3. For each query frame, retrieve the k=10 nearest database frames.
4. For each candidate video, build a histogram of temporal offsets between query and database frames.
5. Select the offset with the highest count as the best alignment.
6. Rank candidate videos by their peak histogram value and return the top-10.

5.3 Evaluation Protocol and Metrics

We evaluate all methods using the following metrics, computed over the full set of queries:

- Mean Reciprocal Rank (MRR): Measures the average inverse rank of the first correct result.
- Mean Average Precision at 10 (mAP@10): Average precision over the top-10 retrieved results.
- Recall@10: Fraction of relevant items retrieved in the top-10.
- Found-Original: Fraction of queries where the original (ground-truth) video is present in the top-10.

All experiments are run with fixed parameters (k=10 nearest neighbors, top-10 returned).

Unfortunately, other solutions for Partial Video Copy Detection haven't measured these metrics and execution speed. Additionally, all these methods either don't have open-sourced code or are closed solutions, which makes it difficult to test them.

6. Results

6.1 Accuracy and Robustness

Fig. 2 and Table 2 summarize the global accuracy metrics for different perceptual hashing algorithms in the Shazam-based video copy detection pipeline. The results include Mean Reciprocal Rank (MRR), mean Average Precision (mAP), recall, and the fraction of queries where the original was found. All tested hash functions demonstrate competitive performance, with most achieving recall and “found original” rates above 0.9. The differences among methods are minor, indicating that even computationally efficient hashes can provide strong detection accuracy. This supports the method’s suitability for large-scale deployments where both speed and accuracy are required.

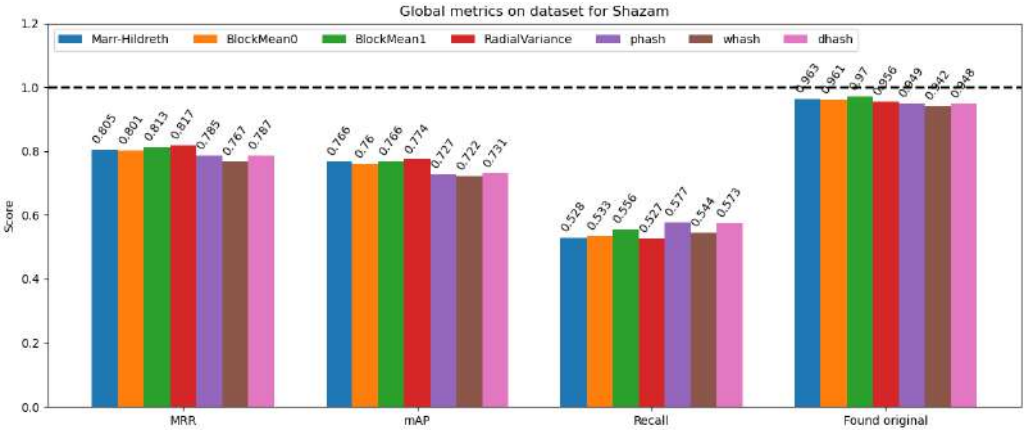


Fig.2. Main metrics across different hash algorithms.

Table 2. Results of accuracy of different hash functions. The best result is presented in bold text.

Hash Name	MRR	mAP	Recall	Found original
Marr-Hildreth	0.805	0.766	0.528	0.963
BlockMean0	0.801	0.760	0.533	0.961
BlockMean1	0.813	0.766	0.556	**0.970**
RadialVariance	**0.817**	**0.774**	0.527	0.956
phash	0.785	0.727	**0.577**	0.949
whash	0.767	0.722	0.544	0.942
dhash	0.787	0.731	0.573	0.948

6.2 Execution Speed and Scalability

The pipeline is optimized for parallel execution. With 14 parallel processes, database construction and query search achieved a 2.9–3.0× speedup compared to single-process operation. Fig. 3 and Table 3 present the average search and hash computation times for different perceptual hashing algorithms in seconds per one-hour video. The orange bars represent the mean hash computation time per algorithm, while the blue bars show the mean search time required to match video fragments using the Shazam-inspired method. Notably, algorithms such as Marr-Hildreth and whash exhibit the highest hash computation times (over 70 and 80 seconds, respectively), whereas simpler hash functions like BlockMean0 and BlockMean1 achieve significantly faster hashing performance. Across all algorithms, mean search time remains low (generally below 7 seconds), confirming the scalability and efficiency of the search stage regardless of hash type. Also, Key Frame Extractor (KFE) executed in around 450 seconds. This demonstrates that the proposed system can efficiently process long videos and large datasets, with the primary computational bottleneck attributable to the choice of hash function rather than the search phase.

7. Future Work

To address the above limitations and further enhance the system's capabilities, future work will focus on:

- **Integration of Learning-Based Features.** Augmenting the pipeline with deep neural descriptors or transformers trained for video similarity may improve robustness to challenging transformations and nontrivial copies.
- **Multimodal Fusion.** Combining visual fingerprints with audio, text, or metadata-based signatures to reduce ambiguity and improve detection in noisy or complex scenarios.

- **Adaptive Parameter Selection.** Developing automatic tuning or meta-learning strategies to select optimal parameters based on data characteristics or operational constraints.
- **Advanced Indexing and Distributed Search.** Exploring scalable indexing techniques, such as hierarchical vector search or distributed hash tables, to support petabyte-scale video archives.
- **Real-World Deployment and User Studies.** Collaborating with industry and the research community to deploy the system in practical applications, gather user feedback, and refine the approach for diverse real-world environments.

By addressing these directions, the method can become an even more powerful tool for the global community, facilitating responsible media management, copyright protection, and digital trust.

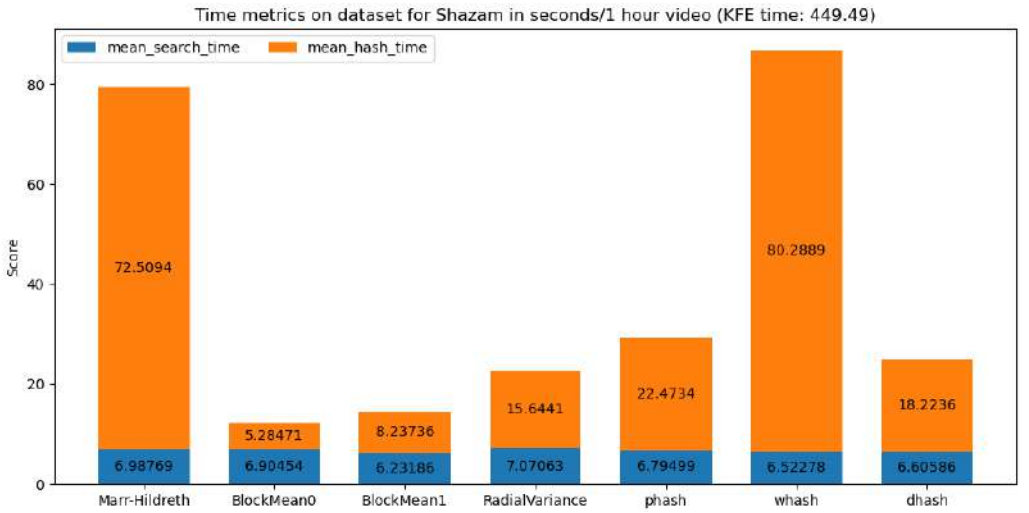


Fig.3. Speed execution of different hash algorithms in seconds per 1 hour video (on 28-core CPU).

Table 3. Results of speed comparison of different hash algorithms. The best result is presented in bold text.

Hash Name	Mean Search Time (s)	Mean Hash Time (s)
Marr-Hildreth	6.98769	72.5094
BlockMean0	**5.28471**	**6.09451**
BlockMean1	8.23796	6.23136
RadialVariance	7.07063	15.6441
phash	6.79499	22.4734
whash	6.52278	80.2889
dhash	6.05098	18.2236

8. Conclusion

In this work, we have introduced a novel method for partial video copy detection, inspired by the proven principles of the Shazam algorithm [1] in audio identification. Our approach combines systematic keyframe extraction, robust perceptual hashing, and efficient search and temporal alignment to address the challenges of detecting copied video fragments within large-scale collections. Code is available as open-source solution [16].

Extensive evaluation on the VCDB benchmark [2] confirms that the proposed method achieves high accuracy, robustness to common video transformations, and real-world scalability through CPU-parallel processing. The pipeline remains accessible and reproducible, relying solely on open-source

tools and standard hardware, making it suitable for broad adoption in scientific, industrial, and societal contexts.

By enabling reliable detection and localization of video copies, this work contributes a transparent and effective solution for content protection, digital rights management, and responsible media use– serving the needs of all humanity in the evolving digital landscape. Future research will explore further improvements in robustness, scalability, and multimodal integration to extend the capabilities of this framework.

References

[1]. Wang A. et al. An industrial strength audio search algorithm // Ismir. 2003. Vol. 2003, pp. 7-13.
[2]. Jiang Y. G., Jiang Y., Wang J. VCDB: a large-scale database for partial copy detection in videos // Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13. Springer International Publishing, 2014, pp. 357-371.
[3]. Chen W., Gan W., Yu P. S. Digital Fingerprinting on Multimedia: A Survey // arXiv preprint arXiv:2408.14155. 2024.
[4]. Pizzi E. et al. The 2023 video similarity dataset and challenge // Computer Vision and Image Understanding. – 2024. Vol. 243, pp. 103997.
[5]. Vaswani A. et al. Attention is all you need // Advances in neural information processing systems. – 2017. Vol. 30.
[6]. Lu J. Video fingerprinting for copy identification: from research to industry applications // Media Forensics and Security. 2009. Vol. 7254, pp. 725402.
[7]. Hua X. S., Chen X., Zhang H. J. Robust video signature based on ordinal measure // 2004 International Conference on Image Processing, 2004. ICIP'04. IEEE, 2004. Vol. 1, pp. 685-688.
[8]. Malekesmaeili M., Fatourechi M., Ward R. K. Video copy detection using temporally informative representative images // 2009 International Conference on Machine Learning and Applications. IEEE, 2009, pp. 69-74.
[9]. Bradski G. The opencv library // Dr. Dobb's Journal: Software Tools for the Professional Programmer. 2000. Vol. 25, issue 11, pp. 120-123.
[10]. Buchner J. Image Hash library [Online] https://github.com/jgraving/imagehash (accessed 20.04.2025).
[11]. Jain T. et al. Imagededup [Online] https://github.com/idealo/imagededup (accessed 20.04.2025).
[12]. Mahanty A. Videohash [Online] https://github.com/akamhy/videohash (accessed 20.04.2025).
[13]. Katiyar A., Weissman J. {ViDeDup}: An {Application-Aware} Framework for Video De-duplication // 3rd Workshop on Hot Topics in Storage and File Systems (HotStorage 11). 2011.
[14]. Steinebach M. An analysis of photodna // Proceedings of the 18th International Conference on Availability, Reliability and Security. 2023, pp. 1-8.
[15]. Liu Z. et al. A similarity alignment model for video copy segment matching // arXiv preprint arXiv:2305.15679. 2023.
[16]. https://github.com/SUPERustam/frame_video_search (accessed 25.08.2025).

Информация об авторах / Information about authors

Рустам Шамилевич УЗДЕНОВ является студентом факультета Фундаментальных Наук МГТУ им. Баумана, лаборант института системного программирования РАН. Научные интересы: нейросетевая обработка данных, цифровая обработка изображений, методы доверенного искусственного интеллекта.

Rustam Shamilevich UZDENOV – a undergraduate student of the Faculty of Fundamental Sciences at Bauman Moscow State Technical University, and a laboratory assistant at the Institute of System Programming of the RAS. Research interests: neural network data processing, digital image processing, trust artificial intelligence.

Андрей Игоревич ПЕРМИНОВ – аспирант института системного программирования РАН. Научные интересы: нейросетевая обработка данных, цифровая обработка изображений, методы доверенного искусственного интеллекта.

Andrey Igorevich PERMINOV – a postgraduate student at the Institute of System Programming of the RAS. Research interests: neural network data processing, digital image processing, trust artificial intelligence.