DOI: 10.15514/ISPRAS-2025-37(2)-8

Усовершенствованный поиск архитектур в автоматическом решении задач графового машинного обучения: расширение и динамическая приоритизация пространства поиска для повышения эффективности

[1, 2] *Ф.М. Балабанов, ORCID: 0009-0005-7510-7659 <fedor.balabanov@list.ru>*
[1, 3, 4] *К.С. Лукьянов, ORCID: 0009-0009-5235-2175 <lukyanov.k@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.*
[3] *Московский физико-технический институт (национальный исследовательский университет), Россия 117303, Москва, ул. Керченская, д.1 А, корп. 1.*
[4] *Исследовательский центр доверенного искусственного интеллекта ИСП РАН, Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.*

Аннотация. В данной работе исследуются методы улучшения процесса автоматизированного поиска архитектур для графовых нейронных сетей (ГНС). Мы предлагаем новый подход, основанный на адаптивном изменении пространства поиска посредством выделения приоритетных направлений, что позволяет повысить эффективность поиска архитектур. Еще один предлагаемый подход расширяет пространство поиска, разрешая комбинировать различные типы графовых сверточных слоев. Основное внимание уделяется максимизации точности архитектур в расширенном пространстве поиска при фиксированном бюджете поиска по количеству моделей. Наши эксперименты проводятся на наборах данных цитирования, химических молекул и графов покупок. Результаты экспериментов показывают, что предложенный подход позволяет находить более эффективные модели без увеличения вычислительных ресурсов и демонстрирует высокую перспективность для автоматизации решений реальных задачах анализа графовых данных.

Ключевые слова: графовая нейронная сеть; поиск архитектур нейросетей; обучение с подкреплением.

Для цитирования: Балабанов Ф.М., Лукьянов К.С. Усовершенствованный поиск архитектур в автоматическом решении задач графового машинного обучения: расширение и динамическая приоритизация пространства поиска для повышения эффективности. Труды ИСП РАН, том 37, вып. 2, 2025 г., стр. 115–128. DOI: 10.15514/ISPRAS–2025–37(2)–8.

Improved Search in Graph AutoML: Expansion and Dynamic Prioritization in the Search Space for Enhanced Efficiency

[1, 2] *F.M. Balabanov, ORCID: 0009-0005-7510-7659 <fedor.balabanov@list.ru>*
[1, 3, 4] *K.S. Lukyanov, ORCID: 0009-0009-5235-2175 <lukyanov.k@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, A. 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), building 1, 1 A, Kerchenskaya st., Moscow, 117303, Russia.*
[4] *Research Center for Trusted Artificial Intelligence ISP RAS, 25, A. Solzhenitsyn st., Moscow, 109004, Russia.*

Abstract. This paper explores methods for enhancing the automated architecture search process for graph neural networks. We propose a novel approach that dynamically selects a priority direction within the search space, improving the efficiency and quality of the discovered architectures. Another proposed approach expands the search space by allowing combinations of different types of graph convolutional layers. The primary focus is on maximizing the quality of architectures within the expanded search space while maintaining a fixed search budget in terms of the number of models. Our experiments are conducted on datasets from citation networks, chemical molecules, and shopping graph domains. The experimental results show that the proposed approach enables the discovery of more effective and higher-quality models without increasing computational resources, demonstrating high potential for automating solutions to real-world graph data analysis tasks.

Keywords: graph neural network; AutoML; neural architecture search; reinforcement learning.

For citation: Balabanov F.M., Lukyanov K.S. Improved search in graph AutoML: expansion and dynamic prioritization in the search space for enhanced efficiency. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 2, 2025, pp. 115-128 (in Russian). DOI: 10.15514/ISPRAS-2025-37(2)-8.

1. Введение

Графовые нейронные сети (ГНС) – это мощный и активно развивающийся класс моделей машинного обучения, который напрямую работает с графовыми структурами данных. Графы как структура данных обладают уникальными свойствами, такими как наличие вершин и рёбер, что позволяет моделировать сложные взаимосвязи между объектами. Применение графов встречается в самых разных областях, включая анализ социальных сетей [1-2], моделирование взаимодействий молекул в химии [3-4], а также в задачах, связанных с анализом научных данных, например, на основе графов цитирования научных статей [5-6]. Эти задачи требуют извлечения глубоких и комплексных закономерностей из данных, что делает графовые нейронные сети важным инструментом для решения широкого спектра задач, где традиционные методы машинного обучения могут быть менее эффективными.

Однако разработка графовых нейронных сетей является нетривиальной задачей, требующей глубокого понимания как специфики графов, так и принципов построения нейронных сетей. Подбор архитектуры нейронной сети, а также оптимизация гиперпараметров являются ключевыми аспектами, которые напрямую влияют на качество модели. Этот процесс требует значительных временных и вычислительных ресурсов, а также высокого уровня экспертизы в области машинного обучения. В связи с этим, автоматизация процесса проектирования нейронных сетей с использованием методов автоматического поиска моделей машинного обучения AutoML (Automated Machine Learning) становится всё более актуальной.

Алгоритмы AutoML [7] предлагает решение этой проблемы, позволяя автоматизировать подбор архитектуры моделей и их гиперпараметров. Использование AutoML значительно снижает порог входа для специалистов из различных областей, предоставляя инструменты

для автоматического проектирования моделей, что особенно полезно в контексте графовых данных. Одним из главных направлений AutoML является поиск архитектур нейросетей (Neural Architecture Search, NAS), что позволяет находить оптимальные архитектуры моделей на основе данных без необходимости ручного подбора и экспериментов. NAS методы часто используют подходы обучения с подкреплением для поиска архитектур, что позволяет моделям "изучать" пространство возможных архитектур на основе сигналов обратной связи, получаемых от качеств обучаемых моделей.

Тем не менее, несмотря на значительные успехи в области NAS, существует несколько серьёзных проблем, связанных с применением этих методов применительно к графовым данным. Во-первых, пространство возможных архитектур для графовых нейронных сетей значительно шире и сложнее, чем для обычных нейронных сетей, что усложняет задачу поиска. Во-вторых, высокая вычислительная стоимость оценки производительности каждой архитектуры делает процесс поиска чрезвычайно затратным по времени и ресурсам. В результате, ускорение процесса NAS для графовых нейронных сетей является актуальной и важной научной задачей.

Целью данной работы является исследование способов ускорения процесса автоматического поиска архитектур графовых нейронных сетей при расширении пространства поиска. Мы предлагаем новых подход выделения приоритетного направления поиска, путем приоритизации свёрточных слоёв, которые себя хорошо зарекомендовали. Исследуется влияние использования различных свёрточных методов в одной архитектуре при поиске архитектур с использованием обучения с подкреплением.

В разделе 2 рассматриваются существующие методы AutoML для графовых нейронных сетей, проводится их анализ и сравнение. В разделах 3 и 4 описываются предложенные методы ускорения поиска архитектур, а также приводятся результаты экспериментов и их анализ. Наконец, в разделе 5 подводятся итоги работы и делаются выводы относительно эффективности предложенных подходов.

Наш вклад можно представить следующим образом:

1) Мы предлагаем новый подход расширения пространства поиска за счёт комбинирования различных типов сверток на разных слоях модели;
2) Мы предлагаем новый подход выделения приоритетного направления, которое может быть зафиксировано или динамически определяться по ходу работы алгоритма AutoML;
3) Мы показываем эффективность и ограничения применения предложенных подходов на трех доменах и для двух различных постановок задач.

2. Обзор

AutoML изучает методы автоматизированного поиска архитектур моделей машинного обучения, в том числе нейронных сетей [7]. Кроме задачи поиска архитектуры, перед методами AutoML может стоять задача подбора гиперпараметров, задачи предобработки и постобработки данных. Самой сложной задачей для AutoML является случай, когда необходимо управлять всем путем решения задачи: от пред обработки данных, до оптимизации гиперпараметров.

Основными проблемами при разработке методов автоматического поиска архитектур являются:

1) Выбор пространства поиска архитектур и способ его представления. Исследователям приходится искать компромисс между размером пространства поиска и его репрезентативностью.
2) Выбор способа подбора архитектур. Можно выделить два класса методов: не использующие в своей основе машинное обучение и использующие его.

От выбора этих двух составляющих будет зависеть скорость подбора достаточно точной архитектуры, и то насколько точной в сравнении со всеми возможными архитектурами она будет.

AutoML в первую очередь предназначен для неспециалистов в области машинного обучения. Он предлагает методы, которые могут гарантировать, что за выделенное время будет найдена модель, которая будет достаточно хорошо решать поставленную задачу. Например, AutoML применяется в медицинских задачах [8]. Это позволяет ученым в области медицины не погружаться глубоко в принципы создания и проектирования нейронных сетей, но при этом применять в своих исследованиях методы машинного обучения.

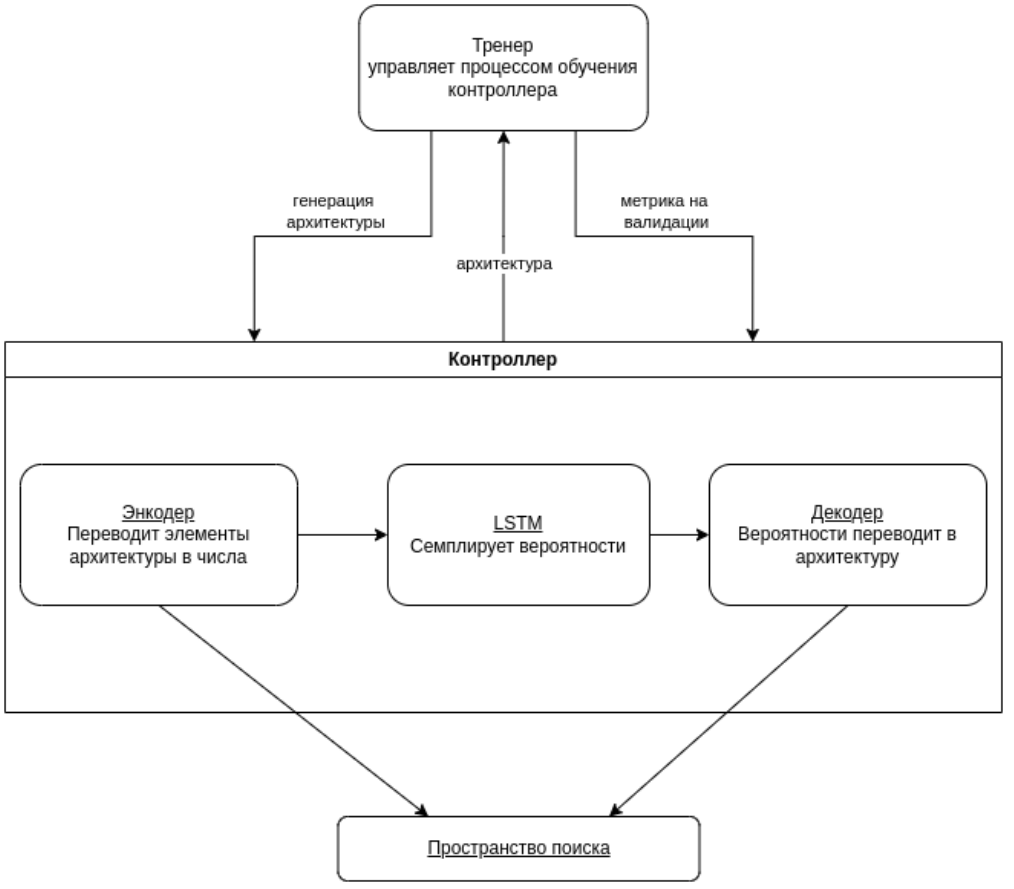

Рис. 1. Архитектура методов NAS, использующих подход обучения с подкреплением.
Fig. 1. Architecture of NAS methods that use a reinforcement learning approach.

Один из подходов к решению задачи NAS использует обучение с подкреплением нейронной сети, генерирующей архитектуры. Изначально такие методы использовались для генерации архитектур свёрточных и реккурентных нейронных сетей [9], но в последствии нашли себе применение и в других областях. Рассмотрим общую концепцию работы всех методов автоматического поиска архитектур нейронных сетей, использующих обучение с подкреплением (рис. 1).

Имеются три главные составляющие:

1) Пространство поиска отвечает за множество возможных вариантов архитектур, которые могут генерироваться контроллером. Выбор пространства поиска важная задача при поиске архитектур нейронных сетей.
2) Тренер отвечает за управление процессом обучения контроллера. Он путем прямого прохода по контроллеру получает архитектуру нейронной сети, измеряет для неё метрику на валидационной выборке и отдает эту метрику обратно контроллеру для обратного прохода.
3) Контроллер нейронной сети состоит из трёх основных частей. Энкодер контроллера отвечает за перевод элементов пространства поиска (возможные свёрточные слои, количество эпох обучения, типы пулинга и т.д.) в числа. Энкодер передает математическое представление пространства поиска рекуррентной нейронной сети (РНС), представленной одним слоем LSTM (Long Short-Term Memory — архитектура нейронной сети долгой краткосрочной памяти) [10]. РНС генерирует список вероятностей для элементов пространства поиска, в соответствии с которыми нужно их выбирать для составления искомой архитектуры нейронной сети. Декодер переводит, полученные от РНС вероятности, в итоговую архитектуру.

В терминах обучения с подкреплением агентом в данном случае является контроллер, окружающей средой – тренер, а наградой – точность архитектуры на валидационной выборке.

Классические свёрточные нейронные сети, работающие с матричными данными, используют один свёрточный метод, в котором варьируется вид ядра и функция агрегирования (рис. 2).

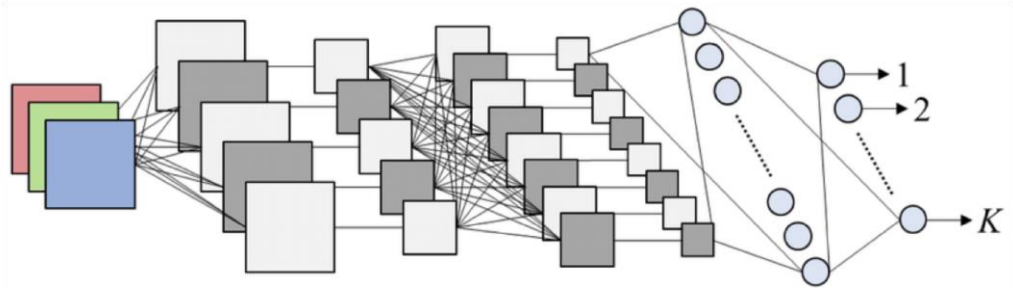

Рис. 2. Принцип работы классических свёрточных нейронных сетей.
Fig. 2. The principle of operation of classical convolutional neural networks.

В то же время, для графовых нейронных сетей (рис. 3) было предложено более 50 различных типов свёрточных слоёв, принципиально отличающихся по принципу своей работы. К тому же каждый свёрточный метод имеет собственный набор гиперпараметров, которые нужно подбирать отдельно для каждой задачи. Указанные отличия в значительной мере увеличивают пространство поиска архитектур ГНС.

GraphNAS [12] является методом NAS, использующим обучение с подкреплением для графовых нейронных сетей. Сложности, которые возникают для метода GraphNAS и которых нет в аналогичных методах при работе с другими типами данных, такие:

1) Графовые данные имеют структуру, которая требует принципиально иного подхода к решению задач;
2) Большое пространство архитектур ГНС.

В качестве награды в обучении с подкреплением GraphNAS использует точность на валидационной выборке. Все архитектуры исследователи представили в обобщенном виде, где каждый слой нейронной сети состоит из следующих функций:

1) функция преобразования внутреннего представления признаков;

2) функция семплирования;
3) функция измерения корреляции вершины с её соседями;
4) функция агрегирования информации в вершине от её соседей;
5) функции, добавляющие к внутреннему представлению информацию с предыдущих слоёв;
6) функции внимания, учитывающие размер соседей.

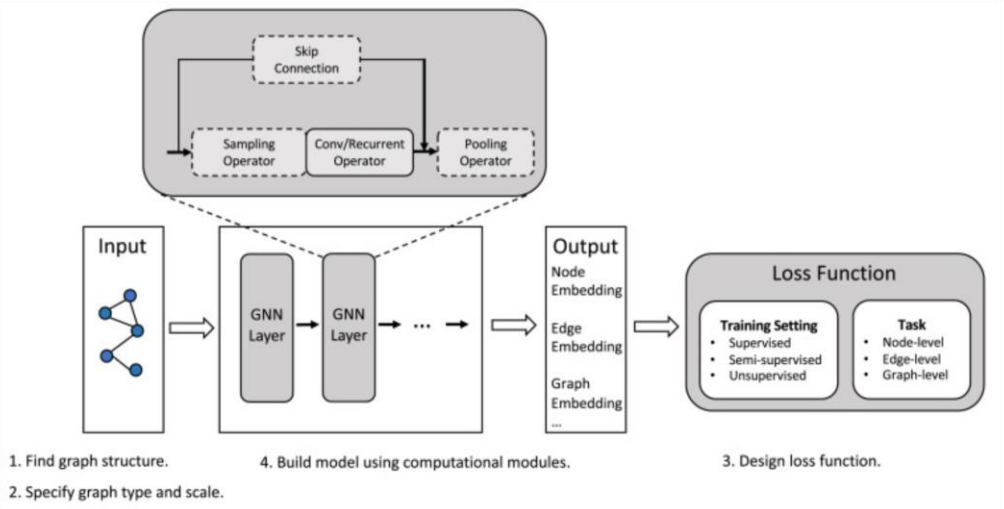

Рис. 3. Архитектура графовых нейросетей из обзорной статьи [11].
Fig. 3. Architecture of graph neural networks from the review article [11].

Авторы статьи [13], разработавшие платформу graphgym, в своей работе нацелены на поиск способа обобщенно подходить к задачам ГНС. Они ввели и рассмотрели общее пространство проектирования ГНС; спроектировали пространство задач на графовых данных с метрикой подобия, позволяющее быстро подбирать архитектуру; разработали эффективный метод оценки пространства поиска. Результатом их работы стал набор рекомендаций по проектированию ГНС, способ переноса лучших архитектур, между похожими задачами. Корреляция между задачами вычислялась путем применения фиксированного набора архитектур к двум задачам и последующего измерения корреляции ранга Кендалла качества этих архитектур. Для поиска архитектур используется контролируемый случайный поиск.

Метод SNAG [14] улучшил методы GraphNAS и Auto-GNN добавив несколько методов ГНС, слои агрегации и уменьшив количество настраиваемых гиперпараметров. Благодаря уменьшению пространства поиска и другому его выбору, авторам удалось достичь лучших результатов.

Авторы метода SANE [15] спроектировали собственное пространство поиска архитектур ГНС, при этом они не оптимизируют гиперпараметры, потому что в ГНС структура имеет большее значение. Это позволило сделать пространство репрезентативным и не слишком большим. Особенностью данной работы является то, что используется дифференцируемая функция награды. Этот метод показал высокое качество работы.

Метод ALGNN [16] нацелен на ускорение процесса подбора модели, с помощью проектирования быстро-обучающихся ГНС. Для поиска архитектур применяют популяционный алгоритм.

Различные методы AutoML для графовых нейронных сетей отличаются выбором пространства поиска: какие обобщённые слои рассматриваются, оптимизируются ли

гиперпараметры – выбором стратегии поиска: случайный поиск, обучение с подкреплением контроллера, генетические алгоритмы, популяционные алгоритмы, дифференцируемые алгоритмы - выбор способа оценки архитектур.

В дальнейшем мы остановимся на ускорении метода GraphNAS, потому что он является базовым для многих других методов AutoML и результаты, полученные на нём, при необходимости, можно будет адаптировать под другие методы.

3. Конфигурация экспериментов

В данном разделе будут зафиксированы и уточнены пространство поиска архитектур ГНС, метрики, применяемые для сравнения методов, и методика проведения экспериментов.

3.1 Реализация системы

Уточним реализацию NAS контроллера и тренера. Как было указано в обзоре, контроллер состоит из энкодера, декодера и одного слоя LSTM. Размерность входа и выхода слоя LSTM равна 100 для всех экспериментов. Для его обучения используется алгоритм оптимизации Adam из библиотеки pytorch с гиперпараметрами по умолчанию.

В качестве награды в процессе обучения с подкреплением тренер использует точность на валидационной выборке для сгенерированной архитектуры. Обратный проход по контроллеру тренер запускает следующим образом:

$$b = b \cdot 0.95 + r \cdot 0.05$$

$$l = -\sum_i logit\, s_i \cdot (r - scale(b, 0.5))$$

где r – значение награды для сгенерированной архитектуры,

b – обновляемое на каждой итерации обучения контроллера значение,

$logits$ – выход логистического преобразования вероятностей, которые генерирует контроллер,

$scale(x, y)$ – функция, переводящая x в отрезок $[-y, y]$, где максимум и минимум соответствуют максимальному и минимальному значению награды для последних 10 сгенерированных архитектур.

3.2 Пространство поиска

Пространство поиска выбрано следующим:

1) Методы ГНС: GCN [19], SAGE [20], GAT [21], TAG [22], GIN [23], SG [24], SSG [25], GMM [26]. Были выбраны представленные методы, потому что они являются широко-используемыми и представленны в библиотеке pytorch [17].
2) Количество эпох обучения для задач классификации графов - 100 - 220 с шагом 20. Количество эпох обучения для задач классификации вершин - 3000 - 8000 с шагом 500. Варианты количества эпох обучения были выбраны в соответствии с проведенными первичными тестами. В этих пределах ГНС для соответствующих задач дают стабильный результат и выходят на свою максимальную точность.
3) Методы пулинга для задач классификации графов: add, mean, max. Эти методы популярные и широко-используемые.

Другие гиперпараметры были фиксированы для всех генерируемых архитектур:

1) Алгоритм оптимизации Adam с параметрами по умолчанию из библиотеки pytorch;
2) Функция активации между всеми слоями - ReLu, а после последнего слоя - LogSoftmax;

3) Для гиперпараметров свёрточных слоёв использовались значения по умолчанию, заданные в библиотеке pytorch-geometric [18]. Для многих задач они являются самыми оптимальными.

Точность ГНС определяется в первую очередь архитектурой и в исследованиях с нашими доменами используют 2-3 графовых свёрточных слоя, поэтому в наших экспериментах архитектуры состоят из двух свёрточных слоёв.

3.3 Коллекции данных

Измерения проводились на коллекциях данных, представленных в табл. 1.

Табл. 1. Параметры коллекций данных. Количество вершин и ребер указано в среднем на граф.
Table 1. Parameters of data collections. The number of vertices and edges is indicated on average per graph.

Данные	Графы	Вершины	Рёбра	Классы	Признаки
Задача классификации вершин на данных о цитировании. Группа данных **Planetoid**.					
Cora	1	2708	10556	7	1433
CiteSeer	1	3327	9104	6	3703
PubMed	1	19717	88648	3	500
Задача классификации вершин на данных о покупках. Группа данных **Amazon**.					
Computers	1	13752	491722	10	767
Photo	1	7650	238162	8	745
Задача классификации графов химических молекул. Группа данных **TuDataset**.					
MUTAG	188	17.9	39.6	2	7
COX2	467	41.22	43.45	2	3
BZR	405	35.75	38.36	2	3

3.4 Метрики

Для сравнения эффективности поиска различных конфигураций, используется метрика показывающая, максимальную точность на валидационной выборке среди архитектур, сгенерированных методом AutoML.

3.5 Методика проведения экспериментов

Полная конфигурация экспериментов задается через yaml файл, в котором указывается:

1) Коллекция данных, с которой проводится эксперимент;
2) Изменять или нет вероятности выбора целевых методов;
3) Использовать различные методы ГНС в одной архитектуре или нет;
4) Количество итераций поиска архитектур;

Для коллекций данных проводились эксперименты с различными конфигурациями контроллеров. Метрики во время обучения собирались в логах и затем анализировались. Для каждой конфигурации эксперимента запускалось 600 итераций поиска архитектур.

4. Метод

Для проверки, работают ли целевые методы лучше остальных, была реализована система предоставления этим методам приоритета при подборе типов свёрточных слоёв. Система имеет три варианта поведения.

Первый вариант (basic) является реализацией оригинального варианта алгоритма, используемого авторами статьи [12], и для нас он является базовым решением для алгоритмов, не использующих сочетания различных свёрточных методов в одной архитектуре. Он не приоритизирует никакие методы, то есть шансы выбрать каждый из методов ГНС в пространстве поиска равны. Список возможных свёрточных методов, из которых выбирает контроллер, следующий: gin, gcn, sg, ssg, gat, gmm, tag, sage.

Второй вариант (fixed) приоритизации фиксирует увеличение вероятности получения целевых методов при случайном выборе из пространства поиска на весь процесс обучения контроллера NAS. Рассматривалось увеличение вероятности в 4 раза, то есть при случайном выборе свёрточных методов из пространства поиска, целевые методы генерируются из него в 4 раза чаще.

Третий вариант (dynamic) динамически изменяет вероятность получения целевых методов при случайном выборе из пространства поиска по время обучения. На заданной итерации обучения контроллера, он перестает увеличивать вероятность получения целевых методов и делает все методы равновероятными. Для генерации архитектуры контроллер использует вероятности, полученные из РНС. Для обучения РНС контроллер использует вероятности после логистического преобразования. На выбранной итерации обучения контроллер начинает обнулять либо только вероятности, либо вероятности и выходы логистического преобразования вероятностей вместе. Поведение контроллера задается конфигурацией эксперимента.

Два варианта алгоритма работы третьего варианта:

$C -$ функция семплирования архитектуры контроллером;

$mainIndexes$и$dupIndexes$ –отвечают за индексы в $SearchSpaceGnn$ соответствующие первому вхождению методов в список и повторным вхождениям методов соответственно, где $SearchSpaceGnn$ – список возможных вариантов свёрточных методов для данной конфигурации эксперимента.

---

*Вариант, изменяющий только вероятности и не влияющий на обучение (**dynamic-probsonly**):*

$logits \leftarrow \mathrm{C}()$
$logProbs \leftarrow LogSoftmax(logits)$
$softmaxLogits \leftarrow Softmax(logits)$
$for i \in dupIndexes do$
    $softmaxLogits_i = 0$
$endfor$
$probs \leftarrow Softmax(softmaxLogits)$

---

*Вариант также влияющий и на обучения (**dynamic-both**):*

$logits \leftarrow \mathrm{C}()$
$minLogit \leftarrow min_i\{logits_i \vee i \in mainIndexes\}$
$for i \in dupIndexes do$
$\quad logits_i \leftarrow minLogit$
$endfor$
$probs \leftarrow Softmax(logits)$
$logProbs \leftarrow LogSoftmax(logits)$

Для проверки стоит ли использовать несколько свёрточных методов в одной архитектуре или нет, через параметры пространства поиска задавалось, какое поведение нужно выбрать: генерировать один метод ГНС, который затем используется на всех свёрточных слоях, или генерировать для каждого слоя метод отдельно.

5. Полученные результаты

В табл. 2 представлены результаты экспериментов. Для каждой коллекции данных показана максимальная точность среди сгенерированных соответствующим алгоритмом AutoML архитектур. На рис. 4 сравнение вариантов алгоритмов с использованием комбинаций свёрточных методов и без них. Для каждой коллекции данных было подсчитано, сколько раз первое место по максимальной точности среди всех методов AutoML занимали методы, использующие комбинации разных свёрточных слоёв в одной архитектуре, а сколько раз не использующие. Затем эти значения были просуммированы по группам наборов данных, а получившиеся данные были отображены на рис. 4. Для построения диаграммы на рис. 5 отдельно анализировались методы, которые не используют комбинации разных свёрточных методов в одной архитектуре ГНС.

Табл. 2. Сравнительная таблица результатов экспериментов по метрике максимального значения точности, среди сгенерированных методом архитектур. Группа Planetoid: Cora, Citeseer, Pubmed. Группа Amazon: Computers, Photo. Группа TuDataset: COX2, MUTAG, BZR.
Table 2. A comparative table of experimental results on the metric of maximum accuracy, among the architectures generated by the method. Planetoid group: Cora, Citeseer, Pubmed. Amazon group: Computers, Photo. TuDataset group: COX2, MUTAG, BZR.

dataset	basic		fixed		dynamic-probsonly		dynamic-both	
	no-combinations	combinations	no-combinations	combinations	no-combinations	combinations	no-combinations	combinations
cora	0,9061	0,8950	0,8840	0,8895	0,9042	0,8987	0,9024	0,8987
citeseer	0,7508	0,7898	0,7898	0,7613	0,7703	0,7733	0,7718	0,7763
pubmed	0,8694	0,8709	0,8659	0,8661	0,8753	0,8666	0,8722	0,8709
computers	0,9128	0,9059	0,9037	0,9000	0,9004	0,9062	0,9135	0,8989
photo	0,9471	0,9562	0,9516	0,9431	0,9516	0,9510	0,9484	0,9542
cox2	0,8936	0,8723	0,8830	0,9043	0,9149	0,8511	0,9043	0,9362
mutag	0,8974	0,9487	0,9231	0,9231	0,9744	0,9231	0,9744	0,9744
bzr	0,8889	0,9136	0,9012	0,9136	0,8889	0,9259	0,8765	0,8889

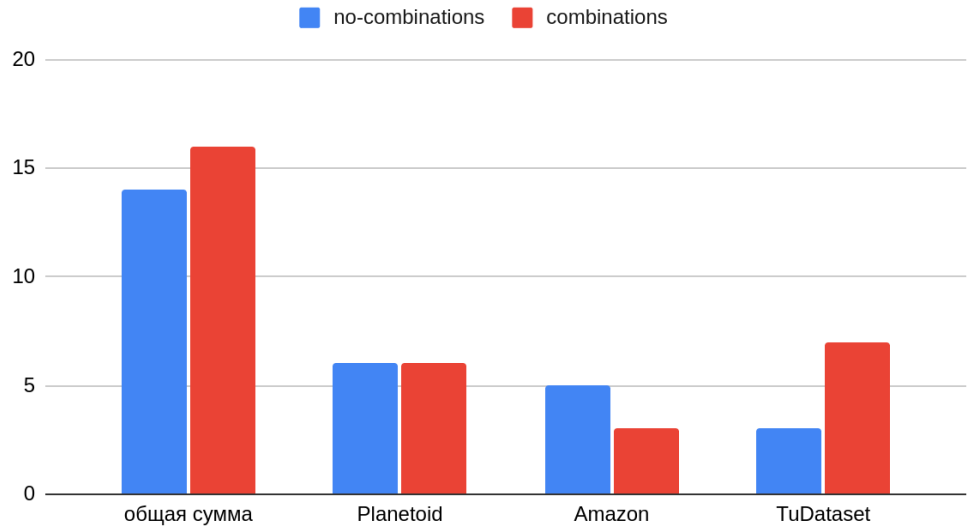

Рис. 4. Каждый столбец – суммарное количество раз, когда алгоритм с комбинациями свёрточных методов (без них), показал результат лучше, чем соответствующий алгоритм без комбинаций (с ними), на каждой из групп наборов данных.
Fig. 4. Each column is the total number of times when the algorithm with combinations of convolutional methods (without combinations) showed a better result than the corresponding algorithm without combinations (with combinations) on each of the dataset groups.

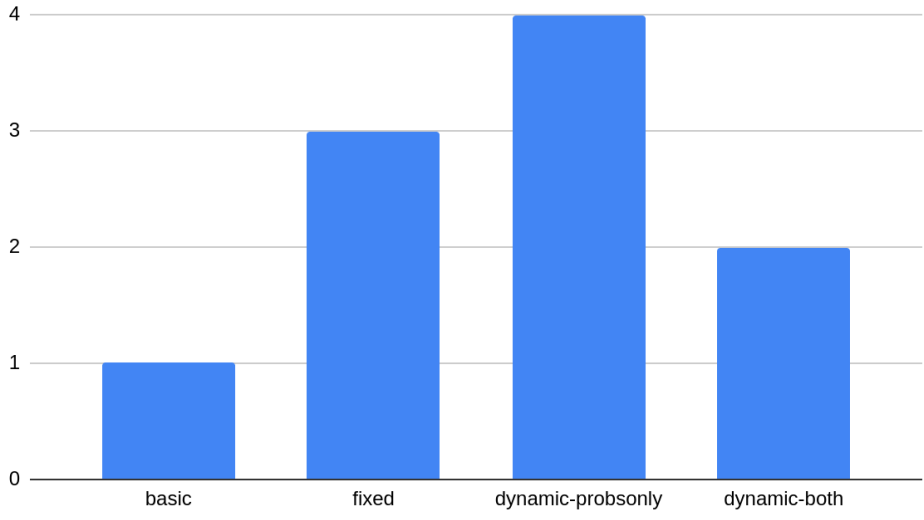

Рис. 5. Сравнение результатов алгоритмов без использования комбинаций свёрточных методов. Значение столбца – количество раз, когда этот метод показал максимальную точность среди других.
Fig. 5. Comparison of results of algorithms without using combinations of convolutional methods. The column value is the number of times this method showed the highest quality among others.

Для каждого метода было посчитано количество коллекций данных, на которых этот метод получил максимальную точность среди других методов, не использовавших комбинации свёрточных методов. Диаграмма на рис. 6 построена аналогично диаграмме на рис. 5, только для методов AutoML, которые использовали комбинации свёрточных слоёв.

Заметим, что для задачи классификации вершин комбинации свёрточных методов в одной архитектуре не улучшили качества, а для задачи классификации графов улучшили. Так же в общем случае динамический вариант алгоритма показывает более эффективные результаты, чем два других.

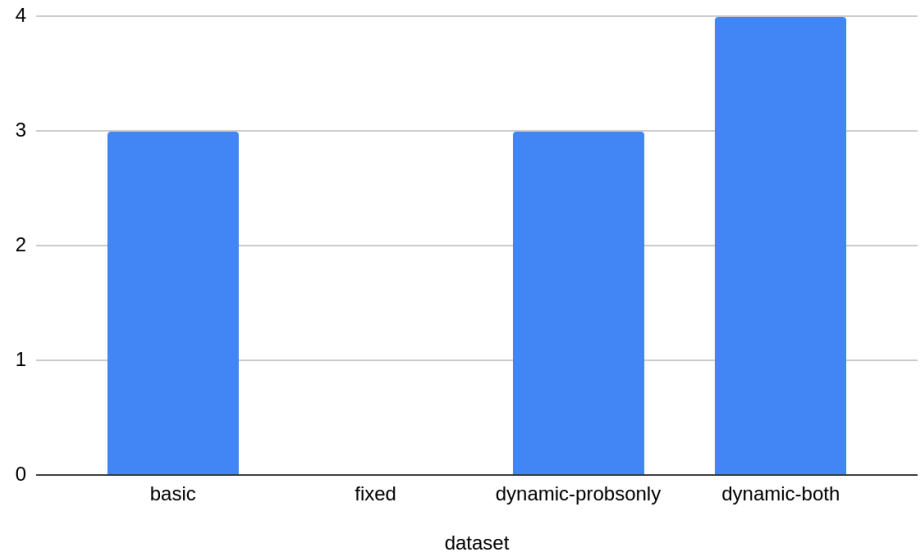

Рис. 6. Сравнение результатов алгоритмов с использованием комбинаций свёрточных методов. Значение столбца - количество раз, когда этот метод показал максимальную точность среди других.

Fig. 6. Comparison of algorithm results using combinations of convolutional methods. The column value is the number of times this method showed the highest quality among others.

6. Ограничения

Обратите внимание, что существенное улучшение подход расширения пространства поиска архитектур за счет разрешения комбинировать разные типы графовых свёрточных слоёв даёт только на данных химического домена. Возможным объяснением этого может быть то, что структура химических молекул располагает к разным типам агрегации.

Также стоит отметить, что использовать фиксированный вариант изменения вероятностей в общем случае менее выгодно, чем динамический подход. Однако, если есть некоторые априорные факторы, указывающие на потенциальную эффективность определённых слоёв, но недостаточный, чтобы полностью исключить возможность выбора других слоёв, то этому подходу может иметь смысл отдать предпочтение.

7. Заключение и будущая работа

В этой статье был предложен новый подход к расширению пространства поиска за счёт комбинирования разных типов свёрточных слоёв при решении задачи поиска архитектуры относящийся к автоматическому машинному обучению для решения задач классификации

вершин и графов. Также был предложен новый подход выделения приоритетного направления в пространстве поиска архитектур и три варианта реализации: фиксированное выделение направления и два динамических варианта выделения направления. Эффективность и ограничения предложенных подходов были продемонстрированы на данных трех доменов и для двух постановок задач.

В качестве направлений будущей работы можно выделить внедрения предложенных подходов для улучшения существующих SOTA решений задачи поиска архитектуры графовых нейросетей, основанных на RL подходе. Также, вероятно, подходы могут быть улучшены посредством использования более современных техник обучения с подкреплением. Еще одним направлением будущих исследований может быть адаптация предложенных подходов для генетических и байесовских подходов решения задачи поиска архитектуры. А также, исследование влияния структурных факторов на результаты работы предложенных подходов, которые приводят к описанным в работе ограничениям.

Список литературы / References

[1]. Xiao Li, Li Sun, Mengjie Ling, Yan Peng A survey of graph neural network based recommendation in social networks. Neurocomputing, 2023, 126441, https://doi.org/10.1016/j.neucom.2023.126441.

[2]. Fan, Wenqi, Ma, Yao, Li, Qing, He, Yuan, Zhao, Eric, Tang, Jiliang, Yin, Dawei Graph Neural Networks for Social Recommendation. 2019, 417–426, 10.1145/3308558.3313488.

[3]. Park, Jaehong, Shim, Youngseon, Lee, Franklin, Rammohan, Aravind, Goyal, Sushmit, Shim, Munbo, Jeong, Changwook, Kim, Dae Sin Prediction and Interpretation of Polymer Properties Using the Graph Convolutional Network. ACS Polymers Au, 2022, 213-222, 10.1021/acspolymersau.1c00050.

[4]. Pietro Bongini, Monica Bianchini, Franco Scarselli Molecular generative Graph Neural Networks for Drug Discovery. Neurocomputing, 2021, 242-252, https://doi.org/10.1016/j.neucom.2021.04.039.

[5]. Fan Zhou, Xovee Xu, Ce Li, Goce Trajcevski, Ting Zhong, Kunpeng Zhang A Heterogeneous Dynamical Graph Neural Networks Approach to Quantify Scientific Impact. 2020.

[6]. Cummings, Daniel, Nassar, Marcel Structured Citation Trend Prediction Using Graph Neural Networks. 2020, 3897-3901, 10.1109/ICASSP40776.2020.9054769.

[7]. Xin He, Kaiyong Zhao, Xiaowen Chu AutoML: A survey of the state-of-the-art. Knowledge-Based Systems, 2021, 106622, https://doi.org/10.1016/j.knosys.2020.106622.

[8]. Rashidi, Hooman H., Tran, Nam, Albahra, Samer, Dang, Luke T. Machine learning in health care and laboratory medicine: General overview of supervised learning and Auto-ML. International Journal of Laboratory Hematology, 2021, 15-22, https://doi.org/10.1111/ijlh.13537.

[9]. Zoph, Barret, Le, Quoc Neural Architecture Search with Reinforcement Learning. 2016.

[10]. Hochreiter, S Long Short-term Memory. Neural Computation MIT-Press, 1997.

[11]. Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, Maosong Sun Graph neural networks: A review of methods and applications. AI Open, 2020, 57-81, https://doi.org/10.1016/j.aiopen.2021.01.001.

[12]. Gao, Yang, Yang, Hong, Zhang, Peng, Zhou, Chuan GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. 2019.

[13]. Jiaxuan You, Rex Ying, Jure Leskovec Design Space for Graph Neural Networks. 2021.

[14]. Zhao, Huan, Wei, Lanning, Yao, Quanming Simplifying Architecture Search for Graph Neural Network. 2020.

[15]. Zhao, Huan, Yao, Quanming, Tu, Weiwei Search to aggregate neighborhood for graph neural network. 2021.

[16]. Cai, Rongshen and Tao, Qian and Tang, Yufei and Shi, Min ALGNN: Auto-Designed Lightweight Graph Neural Network. 2021, 500--512.

[17]. Pytorch library, https://pytorch.org

[18]. Pytorch Geometric library, https://pytorch-geometric.readthedocs.io

[19]. Thomas N. Kipf, Max Welling Semi-Supervised Classification with Graph Convolutional Networks. 2017.

[20]. William L. Hamilton, Rex Ying, Jure Leskovec Inductive Representation Learning on Large Graphs. 2018.

[21]. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio Graph Attention Networks. 2018.

[22]. Jian Du, Shanghang Zhang, Guanhang Wu, Jose M. F. Moura, Soummya Kar Topology Adaptive Graph Convolutional Networks. 2018.

[23]. Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka How Powerful are Graph Neural Networks? 2019.
[24]. Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr. au2, Christopher Fifty, Tao Yu, Kilian Q. Weinberger Simplifying Graph Convolutional Networks. 2019.
[25]. Hao Zhu, Piotr Koniusz Simple Spectral Graph Convolution. 2021.
[26]. Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, Michael M. Bronstein Geometric deep learning on graphs and manifolds using mixture model CNNs. 2016.

Информация об авторах / Information about authors

Фёдор Михайлович БАЛАБАНОВ – лаборант Института Системного Программирования РАН; студент ВМК МГУ. Область научных интересов: графовые нейронные сети, AutoML.

Fedor Mikhailovich BALABANOV – laboratory assistant in the Ivannikov Institute for System Programming of the Russian Academy of Sciences, student at the Moscow State University. His research interests are graph neural networks, AutoML.

Кирилл Сергеевич ЛУКЬЯНОВ – исследователь центра доверенного искусственного интеллекта ИСП РАН; аспирант МФТИ. Область научных интересов: исследования в области доверенного искусственного интеллекта, исследования на пересечении при одновременном обеспечении нескольких критериев доверия (в частности, обеспечение одновременной интерпретируемости и защищенности моделей искусственного интеллекта), AutoML, домены данных – графы, изображения, временные ряды, табличные данные.

Kirill Sergeevich LUKYANOV – Researcher at the Center for Trusted Artificial Intelligence of the Ivannikov Institute for System Programming of the Russian Academy of Sciences; postgraduate student at Moscow Institute of Physics and Technology. Research interests: trustworthy artificial intelligence with a particular focus on studies at the intersection of multiple trust criteria being ensured simultaneously (e.g., achieving both interpretability and robustness of AI models), AutoML, data domains – graphs, images, time series, tabular data.