Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)

Физтех-школа прикладной математики и информатики
Кафедра дискретной математики

Выпускная квалификационная работа бакалавра

Fine-Tuning как эффективная альтернатива регрессии для трансформеров молекулярных структур

Автор:
Студент 025 группы
Орлов Алексей Алексеевич

Научный руководитель:
канд. физ.-мат. наук
Безносиков Александр Николаевич

Москва 2024

Аннотация

Fine-Tuning для трансформеров молекулярных структур

Орлов Алексей Алексеевич

Использование моделей машинного обучения открывает широкие перспективы для эффективного прогнозирования свойств химических соединений. Однако получение размеченных данных о молекулах может занимать много времени и требовать привлечения серьёзных ресурсов. В связи с этим, использование стандартного подхода обучения с учителем может быть затруднено, поэтому в хемоинформатике достаточно часто применяются подходы, основанные на обучении без учителя – например, некоторые языковые (ChemBERTa [1], mol2vec [2]), графовые модели (MolCLR [3]). Логичным продолжением является совмещение достоинств этих двух подходов, что и было проделано [4]. В данной работе применяется аналогичный подход, используя ECFP [5] представление молекул и графовое представление. При работе с химическими соединениями люди часто сталкиваются с ситуацией, когда необходимо дообучение тяжёлой предобученной модели на небольшом узкоспециализированном датасете. Данная работа позволяет решить и эту задачу.

Содержание

1 Введение

1.1 Введение в область

История развития методов машинного обучения начинается с середины 20-го века с появления первых алгоритмов, таких как линейная регрессия и метод наименьших квадратов. Со временем методы усложнялись, и в 1980-х и 1990-х годах были разработаны нейронные сети и методы обратного распространения ошибки, что заложило основу для глубокого обучения. С начала 2000-х годов рост вычислительных мощностей и доступность больших данных ускорили прогресс в машинном обучении. Появление графических процессоров (GPU) и специализированных аппаратных решений позволило обучать глубокие нейронные сети на больших объемах данных, что привело к прорывам в распознавании образов, обработке естественного языка и хемоинформатике. Благодаря своей способности обрабатывать большие объемы данных и выявлять закономерности в молекулярных структурах, введение машинного обучения в хемоинформатике позволило значительно ускорить процесс анализа и предсказания молекулярных свойств, сделав его более эффективным и доступным.

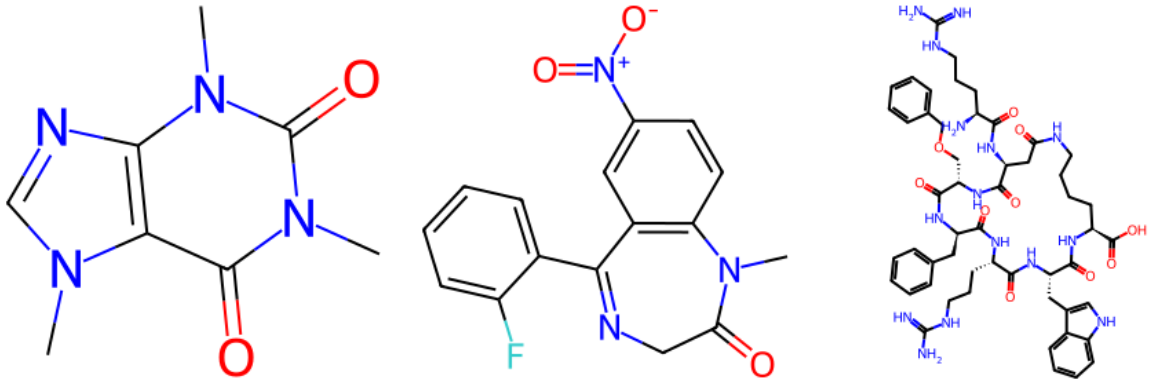
В последние годы методы машинного обучения, особенно те, что основаны на глубоком обучении, демонстрируют значительные успехи в области хемоинформатики и физической химии. Однако ограниченность данных является основной проблемой для обучения моделей, так как каждый экспериментальный результат требует значительных лабораторных усилий. Это ставит химию в контраст с такими областями, как компьютерное зрение или обработка естественного языка, где данные более доступны.

Предсказание свойств молекул является актуальной задачей, поскольку молекулы являются базовыми единицами, на свойствах которых основываются дальнейшие исследования реакций, кристаллов и других химических процессов. Молекулы могут быть представлены в виде нескольких форматов. Например, в виде молекулярного графа, где узлы — это атомы, а рёбра — связи между ними (рис. 1). Этот интуитивно понятный способ визуализации намекает на перспективность использования графовых нейронных сетей (GNN) для анализа и предсказания молекулярных свойств. Очень популярным остается строковое представление химических структур SMILES (Simplified Molecular Input Line Entry System) [6]. Основными преимуществами SMILES являются его простота и удобство в использовании. Также, путём обхода графа в глубину и ряда других правил молекула может быть представлена в виде так называемого молекулярного фингерпринта (FP). В данной работе используется улучшенная версия Morgan Finger Print под названием ECFP (Extended-Connectivity Fingerprints) [5]. Эти представления позволяют использовать различные модели для анализа и предсказания молекулярных свойств.

1.2 Исследование существующих решений

В рамках исследования существующих методов предсказывания молекулярных свойств и получения эмбеддингов молекул были рассмотрены различные подходы, включая mol2vec [2], ChemBERTa [1], SMILES-BERT [7] и другие. Эти методы основаны на принципах обработки естественного языка и используются для преобразования молекулярных структур в векторные представления, которые могут служить входными данными для алгоритмов машинного обучения.

- Mol2vec [2] — это метод, аналогичный Word2Vec в NLP, который использует необученный подход для получения векторных представлений молекулярных подструктур. Он преобразует молекулы в “предложения” и обучает модель на этих данных,



(a) (b) (c)

Рис. 1: Графовые представления молекул.

что позволяет создавать векторы для новых молекул. Однако, он может столкнуться с ограничениями при работе с более сложными молекулярными структурами.

- ChemBERTa [1] — это модель, которая использует трансформеры для предсказания свойств молекул. Она была одной из первых попыток систематически оценить трансформеры на задачах предсказания молекулярных свойств. ChemBERTa показала хорошие результаты при предварительном обучении на больших наборах данных и предоставила конкурентоспособную производительность на MoleculeNet [8], а также полезные визуализации на основе внимания. Однако, как и другие модели, основанные на трансформерах, она может требовать значительных вычислительных ресурсов.

- SMILES-BERT [7] — это подход, который применяет модели BERT к данным SMILES для дизайна лекарств, химического моделирования и предсказания свойств. Этот подход использует предварительно обученные модели на различных наборах данных, таких как ZINC [9] и PubChem [10], и предоставляет веса модели для использования через библиотеку HuggingFace. Однако, использование SMILES как входных данных может привести к потере структурной информации, так как одна и та же молекула может быть представлена различными SMILES-строками.

Однако, несмотря на их потенциал, эти методы сталкиваются с определёнными ограничениями, особенно при использовании SMILES как входных данных. SMILES содержит только структурную информацию о молекуле, а также, как было упомянуто выше, одна и та же молекула может быть представлена различными SMILES-строками, что может затруднить обучение моделей и снизить точность предсказаний.

Важно отметить, что выбор метода зависит от конкретной задачи и доступных ресурсов. Например, если вам доступны большие вычислительные ресурсы, ChemBERTa может быть хорошим выбором. Если же вам нужно быстро получить векторные представления молекул, Mol2vec может быть более подходящим решением. В любом случае, все эти методы представляют собой важные инструменты в области химического моделирования и предсказания молекулярных свойств.

1.3 Цель работы

Целью данной работы является разработка и тестирование методов дообучения (fine-tuning) моделей трансформеров для предсказания молекулярных свойств на неболь-

ших специализированных выборках. Для достижения этой цели используются современные технологии и модели, такие как RoBERTa и MolCLR.

RoBERTa (Robustly optimized BERT approach) [11] представляет собой улучшенную версию модели BERT, которая демонстрирует высокую эффективность в задачах обработки текста и анализа последовательностей, таких как ECFP. MolCLR (Molecular Contrastive Learning of Representations) [3] использует метод контрастивного обучения, который позволяет моделям лучше понимать взаимосвязи между молекулярными структурами и их свойствами.

Особое внимание в данной работе уделено объединению моделей RoBERTa и MolCLR, что позволяет использовать преимущества обеих технологий. RoBERTa обеспечивает качественную обработку последовательностей, а MolCLR улучшает представление молекулярных структур через контрастивное обучение. Такое объединение позволяет значительно повысить точность предсказаний и улучшить обобщающую способность моделей.

Кроме того, в работе рассматривается возможность замены MolCLR на Graphormer, так как он лучше справляется с предсказанием свойств молекул. Graphormer сочетает преимущества графовых нейронных сетей и трансформеров, что позволяет более точно предсказывать свойства молекул, особенно для сложных молекулярных структур.

В рамках исследования проводится обучение модели Graphormer и сравнение её с MolCLR на задаче регрессии молекулярного свойства Molecular Weight. Это позволяет оценить эффективность Graphormer в сравнении с MolCLR и определить, какой подход является более оптимальным для предсказания молекулярных свойств.

Основной вклад данной работы заключается в предложении нового подхода к обучению моделей с использованием комбинированных представлений молекул, что позволяет повысить точность предсказаний на небольших выборках данных. Это исследование открывает новые возможности для эффективного применения машинного обучения в хемоинформатике и физической химии.

2 Постановка задачи

Работы по созданию эмбеддингов молекул можно разделить на два типа: использующие GNN и трансформеры. Оба подхода имеют ряд недостатков, заметных при использовании для решения прикладных химических задач. Например, при обучении почти никогда не используются физические свойства молекул, которые можно подавать в виде категориальных признаков. Большие модели, такие как BERT и многие типы GNN, достаточно плохо настраиваются под маленькие выборки веществ с специфическими физико-химическими свойствами. Кроме этого, на некоторых специфических молекулах трансформеры работают сильно хуже других (рис. 1 пункт б), а на молекулах со сложной структурой, с большим количеством атомов (рис. 1 пункт в) ошибаются графовые модели.

Задачи исследования:

1. Параллельное обучение эмбеддингов: Используя BERT и GNN, обучить модели на молекулярных фингерпринтах и графовых представлениях с помощью известных подходов. Провалидировать полученные модели на задаче предсказывания физических свойств. Использовать дополнительную модель для управления выходами обеих моделей и применить dual-view consistency подход, который заключается в максимизации сходства между проекциями представлений молекул в проективных пространствах по различным метрикам.
2. Файнтьюнинг моделей: Разработка алгоритма для тонкой настройки моделей под малые выборки с физическими свойствами, включая использование специализированного оптимизатора.

Формальная постановка задачи

Дано:

- Множество молекул $\mathcal{M} = \{m_1, m_2, ..., m_n\}$.
- Функции представления молекул в виде фингерпринтов $f_{FP} : \mathcal{M} \to \mathbb{R}^d$ и графовых структур $f_G : \mathcal{M} \to \mathbb{R}^d$.
- Набор физических свойств $\mathcal{P} = \{p_1, p_2, ..., p_k\}$, соответствующих молекулам из $\mathcal{M}$.

Требуется:

1. Обучить модели BERT и GNN, $\mathcal{B} : \mathbb{R}^d \to \mathbb{R}^h$ и $\mathcal{G} : \mathbb{R}^d \to \mathbb{R}^h$, соответственно, используя известные подходы на представлениях молекул $f_{FP}(m_i)$ и $f_G(m_i)$.
2. Провалидировать модели на задаче предсказания физических свойств $\mathcal{P}$.
3. Использовать дополнительную модель $\mathcal{D}$ для управления выходами моделей BERT и GNN.
4. Применить подход dual-view consistency, который формализуется как задача оптимизации:

$$\max_{\mathcal{B},\mathcal{G}} \sum_{i=1}^{n} \text{sim}\left(\mathcal{D}\left(\mathcal{B}\left(f_{FP}(m_i)\right)\right), \mathcal{D}\left(\mathcal{G}\left(f_G(m_i)\right)\right)\right),$$

где $\text{sim}(\cdot, \cdot)$ — метрика сходства, например, косинусное сходство.

3 Обзор используемых инструментов

3.1 Языковая модель

Языковые модели в машинном обучении играют ключевую роль в обработке и понимании естественного языка. *Языковая модель* — это математическая модель, применяемая для расчета вероятности последовательности слов или фраз в естественном языке, обозначаемой как

$$P(w_1, w_2, ..., w_n)$$

где w_1, w_2,..., w_n представляют собой токены, обычно слова или n-граммы. Вместо вычисления общей вероятности текста, часто используются модели для оценки условной вероятности

$$P(w_n|w_1, w_2, ..., w_{n-1})$$

Эти методы являются взаимозаменяемыми, так как полная вероятность может быть выражена через условные вероятности следующим образом:

$$P(w_1, w_2, ..., w_n) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1, w_2) \cdot ... \cdot P(w_n|w_1, w_2, ..., w_{n-1})$$

Языковые модели лежат в основе множества задач в области обработки естественного языка, включая машинный перевод, автоматическое заполнение, классификацию текстов и другие.

Существуют различные виды языковых моделей, в том числе статистические, такие как скрытые марковские модели (HMM [12]), и основанные на нейронных сетях, например, рекуррентные нейронные сети (RNN [13]) и трансформеры.

Статистические модели оценивают вероятность слов на основе их последовательности, что позволяет использовать их для предсказания следующего слова или классификации текста по тематике. Однако они ограничены в понимании контекста слов, что затрудняет их использование для генерации текста.

Нейросетевые языковые модели обучаются на больших объемах данных и могут более точно предсказывать вероятность токенов, что делает их более эффективными для генерации текста, анализа тональности и других задач обработки естественного языка.

В целом, языковые модели являются ключевым инструментом в анализе и обработке естественного языка, находя важное применение в областях, таких как компьютерная лингвистика, машинное обучение и искусственный интеллект. В данной работе, в качестве токенов будут использоваться числа из последовательности ECFP, которые представляют собой молекулярные отпечатки, позволяющие уникально идентифицировать молекулярные структуры.

В контексте машинного обучения, языковая модель представляет собой алгоритм, способный предсказывать следующий элемент в последовательности данных, основываясь на предыдущих элементах. Для текстовых данных, таких как предложения или отпечатки молекул, это означает возможность предсказывать следующее слово или символ. Модель обучается на большом объеме текстовых данных, анализируя контекст и улавливая зависимости между словами.

3.2 Модель RoBERTa

3.2.1 BPE токенизация

Токенизация - это процесс преобразования текстовых данных в числовой формат, с которым работают методы машинного обучения. Простейший пример токенизации - это разбить текст на части и закодировать каждую часть. Например, можно разбить

текст на слова или символы и каждому уникальному слову или символу присвоить свой номер. Однако у этих подходов есть свои недостатки. При кодировании отдельных символов модели будут работать на уровне символов и будет очень сложно восстановить смысл текста по отдельным символам. Со словами возникает проблема в том, что слово может иметь много различных форм, и чтобы учесть все, нужен будет слишком большой словарь, что отрицательно скажется на производительности.

Эффективным решением данной проблемы является Byte Pair Encoding (BPE) [14] токенизация. Принцип работы этого алгоритма следующий: сначала текст кодируется на уровне символов. Затем итеративно выполняются следующие шаги:

- Выбирается наиболее часто встречающаяся пара токенов
- Эта пара объединяется в новый токен, и все вхождения этой пары в тексте заменяются на новый токен

Процесс повторяется до тех пор, пока размер словаря не достигнет заранее установленного предела. Такой токенизатор обучается на большом корпусе текстов и затем используется для токенизации данных на входе и выходе моделей в процессе их обучения и применения.

3.2.2 Общая архитектура

RoBERTa (Robustly Optimized BERT Pretraining Approach) [11] представляет собой модель, основанную на архитектуре BERT (Bidirectional Encoder Representations from Transformers) [15], разработанную в Facebook. Эта модель использует трансформеры для обработки последовательностей входных данных и генерации представлений слов в предложении, учитывая контекст.

RoBERTa сохраняет основную архитектуру BERT, которая является архитектурой трансформера с механизмом внимания. Математически описать это можно следующим образом:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

где Q, K, V обозначают матрицы запросов, ключей и значений соответственно, а d_k — размерность матрицы ключей.
Основная идея BERT заключается в предварительном обучении глубоких двунаправленных представлений от неаннотированного текста, одновременно учитывая контекст слева и справа. Кроме этого, в отличие от оригинальной архитектуры трансформера, которая включает в себя энкодеры и декодеры, BERT использует только энкодеры, что делает его идеальным для задач понимания языка. Однако, в RoBERTa были внесены изменения в процесс предварительного обучения и ключевые гиперпараметры.

Особенности RoBERTa:

- RoBERTa не использует задачу Next Sentence Prediction при обучении, что улучшает или по крайней мере не ухудшает производительность на задачах нижнего уровня.
- Более длительное обучение с большими размерами батчей, что позволяет модели лучше обобщать и достигатьл лучшей точности.
- Динамическое маскирование: В отличие от BERT, где маскирование выполняется один раз во время предварительной обработки данных, RoBERTa использует динамическую маскирование, что позволяет модели изучать более устойчивые представления.

- Обучение на большем объеме данных: RoBERTa была обучена на наборе данных объемом 160 ГБ, что в десять раз больше, чем набор данных, использованный для обучения BERT.

Схожие модели: CamemBERT и XLM-RoBERTa являются примерами моделей, основанных на RoBERTa, которые, также как и в данной работе, были адаптированы для работы с различными языками.

RoBERTa представляет собой значительное улучшение архитектуры BERT, обеспечивающее более эффективное предварительное обучение и лучшую производительность на широком спектре задач обработки естественного языка.

3.2.3 Модель RobertaForMaskedLM

RobertaForMaskedLM [16] — это вариация модели RoBERTa, предназначенная специально для задачи Masked Language Modeling (MLM). Она отличается от стандартной модели RoBERTa тем, что включает в себя дополнительный слой RobertaLMHead, который используется для предсказания вероятности появления слова на месте маски.

Принцип работы RobertaForMaskedLM представлен на рисунке 2. RobertaLMHead получает векторы скрытых состояний от предыдущих слоёв трансформера. Скрытые состояния проходят через линейный слой, который преобразует их в векторы, размерность которых соответствует размеру словаря. Далее применяется функция softmax для получения распределения вероятностей по словарю (здесь модель предсказывает вероятность каждого слова оказаться на месте маскированного токена).

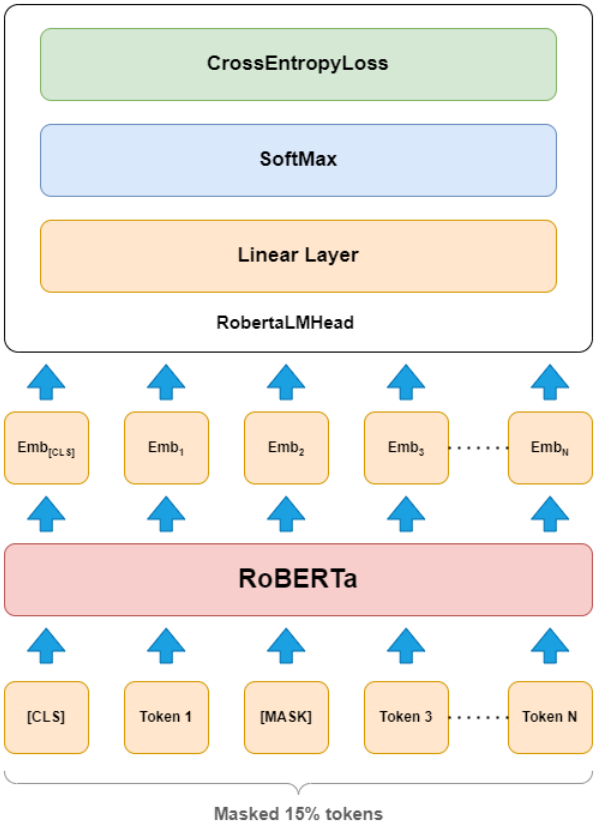


Рис. 2: Принцип работы RoBERTaForMaskedLM.

В дополнительном слое RobertaLMHead используется функция потерь CrossEntropyLoss для вычисления ошибки между предсказанными моделью вероятностями и истинными метками. Это стандартный подход при обучении моделей MLM, таких как RoBERTa.

Функция CrossEntropyLoss вычисляет, насколько предсказания модели отличаются от фактических меток:

$$\text{CrossEntropyLoss}(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i) \tag{2}$$

где y — истинные метки, а $\hat{y}$ — предсказанные вероятности модели, а суммирование происходит по всем классам (словам в словаре). Функция потерь штрафует модель больше, если предсказанная вероятность для истинного класса низкая, и наоборот, меньше штрафует, если вероятность высокая. Это побуждает модель улучшать свои предсказания с каждой итерацией обучения.

Важно отметить, что веса из слоя встраивания (embedding layer), который преобразует входные токены в векторы, используются повторно для линейного преобразования в RobertaLMHead, что позволяет уменьшить общее количество параметров модели и улучшить её обобщающую способность. Это также известно как "weight tying" в контексте языковых моделей.

В контексте MLM это означает, что модель учится предсказывать вероятность каждого слова в словаре для маскированного токена, и функция потерь оценивает, насколько хорошо модель справляется с этой задачей.

3.3 ECFP

3.3.1 Определение

Extended-Connectivity Fingerprints - это тип молекулярных отпечатков, специально предназначенных для выявления молекулярных особенностей, связанных с молекулярной активностью. Они являются одними из самых популярных инструментов поиска сходства при разработке лекарств и эффективно используются в самых разных областях применения.

3.3.2 Алгоритм генерации

1. **Инициализация**: Каждому атому в молекуле присваивается уникальный целочисленный идентификатор, который генерируется путем хеширования комбинации свойств атома.

2. **Обновление**: На каждом этапе итерации алгоритма ECFP, идентификаторы атомов обновляются путем сбора информации от соседних атомов и применения хеш-функции для создания нового уникального идентификатора.

3. **Дедупликация**: Удаляются дублирующиеся особенности из сгенерированного списка особенностей, чтобы каждая уникальная подструктура была представлена однократно.

   Эти шаги повторяются для всех атомов в молекуле, позволяя сформировать массив уникальных числовых идентификаторов, которые представляют различные подструктуры молекулы. После завершения всего процесса каждый атом будет иметь идентификатор, который будет содержать субструктурную информацию со всех частей молекулы. Полученный массив чисел может быть использован для анализа молекулярных структур и моделирования структуры.

3.3.3 Использование в языковой модели

Языковые модели могут быть полезны для работы с последовательностями ECFP (Extended Connectivity Fingerprints), которые являются числовым представлением молекулярных структур. Использование языковых моделей в этом контексте может помочь в предсказании эмбеддингов молекул, что, в свою очередь, может облегчить решение различных задач, связанных с молекулярным моделированием и анализом.

Языковые модели (особенно те, что основаны на нейронных сетях, такие как BERT или Graph Neural Networks) могут изучать сложные зависимости и паттерны в данных. Они способны обрабатывать последовательности токенов и предсказывать вероятности следующих токенов и/или их векторные представления. В случае ECFP, которые кодируют информацию о химической структуре молекул, языковые модели могут использоваться для генерации представлений, которые отражают молекулярные свойства и потенциальные биологические активности.

Таким образом, подавая ECFP в качестве входных данных для языковой модели, можно обучить модель предсказывать эмбеддинги, которые могут быть использованы для различных приложений, таких как поиск похожих молекул, предсказание фармакологических свойств или оптимизация молекулярных структур для улучшения желаемых характеристик. Важно отметить, что успех такого подхода зависит от качества и количества обучающих данных, а также от способности модели обрабатывать специфику молекулярных фингерпринтов. Это может потребовать тонкой настройки и адаптации существующих языковых моделей под конкретные задачи в области химии и молекулярной биологии.

3.3.4 Применение к BPE токенизатору

В рамках исследования был предложен подход к применению алгоритма Byte Pair Encoding (BPE) [14] для токенизации отпечатков ECFP, представленных в виде массива чисел. Достаточно представить имеющиеся отпечатки как строку, где словами будут являться числа из массива молекулы.

Более формально: Зафиксируем молекулу. Пусть $E = \{e_1, e_2, \ldots, e_n\}$ будет её массивом ECFP, где каждый элемент e_i представляет собой целочисленное значение, полученное в результате хеширования молекулярных подструктур. Этот массив преобразуется в строку S, где каждое число e_i представлено как отдельное слово (слова разделены пробелом):

$$S = "e_1\ e_2\ \ldots\ e_n"$$

Далее, к строке S применяется алгоритм BPE, который итеративно заменяет наиболее часто встречающиеся пары чисел на новые токены, уменьшая тем самым размер словаря и обеспечивая компактное представление входных данных для модели.

Экспериментальные результаты показали, что предложенный метод токенизации ECFP с использованием BPE позволяет трансформерам эффективно обрабатывать молекулярные данные, сохраняя при этом значимую информацию о структуре и свойствах молекул.

3.3.5 Сравнение с SMILES

В отличие от SMILES, которые представляют линейную последовательность символов для описания молекул, ECFP генерируют векторы, которые кодируют более сложную информацию о молекуле, включая её топологическое расположение и химические свойства. Это делает ECFP особенно полезными для задач, где требуется улавливать

сложные молекулярные взаимодействия и свойства, таких как предсказание биологической активности или свойств связывания.

3.4 Графовая модель

Графовые нейронные сети (GNN) - это тип искусственных нейронных сетей, предназначенный для работы с данными, структурированными в виде графов. Графы — это математические конструкции, которые используются для представления объектов и их связей, где узлы — объекты, а ребра — связи между ними.

В отличие от обычных нейронных сетей, которые созданы для ввода в виде сетки, GNN работают непосредственно с графами. Они способны изучать представления на уровне узлов и графов, которые полезны для широкого круга задач. Например, GNN можно использовать для задач классификации узлов, целью которых является присвоение метки каждому узлу в графе на основе его характеристик и характеристик его соседей. GNN также можно использовать для задач классификации графов, целью которых является прогнозирование свойств всего графа на основе его структуры и особенностей.

В контексте молекулярных структур, графовые нейронные сети могут быть особенно полезны. Молекулы можно представить в виде графов, где атомы являются узлами, а химические связи - ребрами. GNN могут быть использованы для изучения этих молекулярных графов, предсказания их свойств, и даже для генерации новых потенциальных молекул для лекарственного дизайна. Это делает GNN мощным инструментом в области компьютерной химии и биоинформатики.

3.5 Модель GCN

Graph Convolutional Network (GCN) [17] - это класс графовых нейронных сетей, который использует операцию свертки в контексте графовых структур данных. Основное отличие слоя GCN от линейного слоя заключается в том, что GCN учитывает структуру графа и связи между узлами при обновлении своих весов.

В традиционных нейронных сетях линейные слои применяют линейное преобразование к входным данным. Это преобразование преобразует входные характеристики x в скрытые векторы h с использованием матрицы весов $\mathbf{W}$. Если на время не учитывать смещения, это можно выразить как:

$$h = \mathbf{W}x$$

С данными графа добавляется дополнительный слой сложности через связи между узлами. Эти связи важны, потому что, как правило, в сетях предполагается, что похожие узлы скорее будут связаны друг с другом, чем непохожие, явление, известное как гомофилия сети.

Мы можем обогатить наше представление узла, объединив его характеристики с характеристиками его соседей. Эта операция называется сверткой или агрегацией по соседству. Давайте обозначим окрестность узла i, включая сам узел, как N_i

$$h_i = \sum_{j \in N_i} \mathbf{W}x_j$$

Наши сверточные нейронные сети (CNN) отличаются тем, что у нас есть уникальная матрица весов $\mathbf{W}$, которая используется всеми узлами. Однако мы сталкиваемся с проблемой: количество соседей у узлов не фиксировано, в отличие от пикселей.

Чтобы значения для всех узлов были сопоставимы и находились в одном диапазоне, мы можем нормализовать результат, учитывая степень узлов, где под степенью понимается количество связей узла.

$$h_i = \frac{1}{deg(i)} \sum_{j \in N_i} \mathbf{W} x_j$$

Введенный Кипфом и др. (2016), у слоя графовой свертки есть еще одно улучшение. Авторы заметили, что признаки от узлов с большим количеством соседей распространяются гораздо легче, чем от более изолированных узлов. Чтобы компенсировать этот эффект, они предложили присваивать большие веса признакам от узлов с меньшим количеством соседей, тем самым балансируя влияние по всем узлам. Эта операция записывается как

$$h_i = \sum_{j \in N_i} \frac{1}{\sqrt{deg(i)deg(j)}} \mathbf{W} x_j \tag{3}$$

Существуют различные модификации слоя GCN, которые вносят улучшения или изменения в базовую модель GCN для улучшения производительности или адаптации к конкретным типам задач. Например, некоторые модификации могут включать в себя различные стратегии агрегации информации из соседних узлов или использование различных типов весовых матриц.

В отношении известных моделей, использующих GCN, можно отметить, что GCN является основой для многих других архитектур графовых нейронных сетей. Например, Graph Attention Networks (GAT) [18] и Message Passing Neural Networks (MPNN) [19] являются двумя известными архитектурами, которые используют GCN в качестве основы.

3.6 Модель MolCLR

Molecular Contrastive Learning of Representations (MolCLR) [3] была представлена в работе Юянга Ванга и его коллег в 2022 году. Основная архитектура модели представлена на рисунке 3.

MolCLR использует подход контрастного обучения для обучения представлений молекул на больших неразмеченных наборах данных (порядка 10 миллионов молекул). Этот подход значительно улучшает производительность моделей графовых нейронных сетей на различных задачах прогнозирования свойств молекул.

В основе MolCLR лежит идея, что молекулы, которые имеют схожую структуру или свойства, должны иметь близкие представления в пространстве признаков. Для достижения этого MolCLR использует Contrastive Loss: модель обучается минимизировать расстояние между позитивными парами (то есть молекулами, которые считаются схожими) и максимизировать расстояние между негативными парами (то есть молекулами, которые считаются различными).

В модели MolCLR используются слои GCN [17] или GIN (Graph Isomorphism Network [20]) для кодирования графов молекул. Эти слои графовых нейронных сетей позволяют модели эффективно обрабатывать структурную информацию о молекулах и извлекать из нее полезные признаки.

В данном подходе предлагаются три стратегии аугментации графа молекул: маскирование атомов, удаление связей и удаление подграфов. Эти аугментации применяются к графам молекул, чтобы создать “позитивные” пары (то есть пары аугментаций из одной и той же молекулы), которые модель стремится приблизить в пространстве признаков.

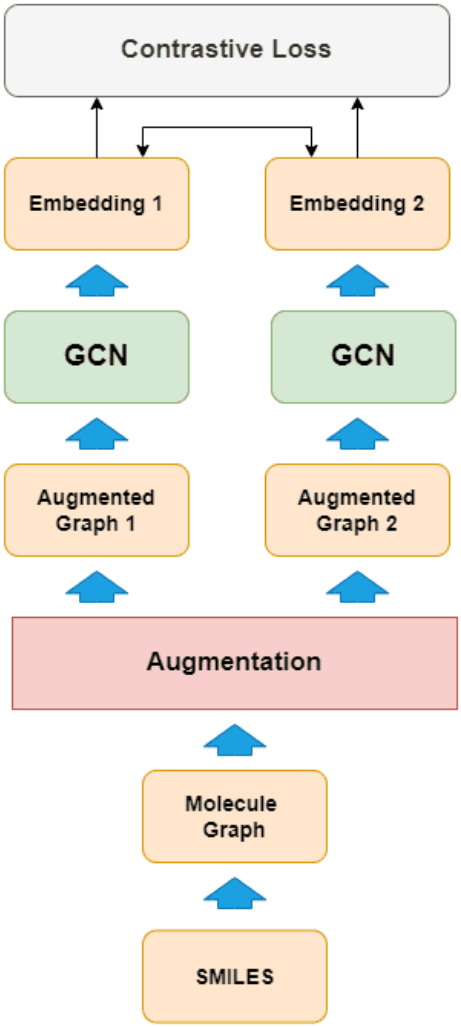


Рис. 3: Архитектура модели MolCLR.

После предварительного обучения с использованием контрастного обучения, MolCLR может быть дообучена на конкретной задаче прогнозирования свойств молекул, такой как прогнозирование растворимости или токсичности.

Важно отметить, что благодаря предварительному обучению на больших неразмеченных наборах данных, MolCLR показывает хорошие результаты на различных задачах прогнозирования свойств молекул.

3.7 Модель Graphormer

Graphormer [21] — это модель глубокого обучения, разработанная Microsoft Research Asia и основанная на архитектуре Transformer, адаптированная для работы с графами. Основное отличие Graphormer заключается в эффективном кодировании структурной информации графа в модель, что достигается за счет использования специальных методов структурного кодирования. Она показала отличные результаты в различных задачах представления графов, особенно в рамках OGB Large-Scale Challenge.

Graphormer может быть полезен по нескольким причинам:

1. Graphormer эффективно кодирует структурную информацию графа, что делает его подходящим для моделирования молекулярных структур.

2. Данная модель показала отличные результаты на широком диапазоне задач обучения представлению графов.

3. Graphormer может быть использован для обучения пользовательских моделей для задач моделирования молекул.

4. Graphormer может покрывать многие популярные варианты GNN в качестве своих специальных случаев.

Для подготовки данных к обучению Graphormer необходимо использовать специальную функцию collate, которая группирует признаки нескольких графов в батчи. Это позволяет эффективно подавать данные в модель. Предобработка включает в себя преобразование данных в формат, совместимый с Graphormer, включая создание объектов DGLGraph и использование стандартного PyTorch Dataloader для подачи данных в модель.

Обучение Graphormer включает в себя стандартные этапы, такие как маскирование части входных данных и предсказание вероятностей для замаскированных элементов. Важно отметить, что модель может неэффективно работать с очень большими графами (более 100 вершин), так как это может привести к переполнению памяти. В таких случаях рекомендуется уменьшить размер батча или использовать более мощные вычислительные ресурсы.

3.8 Графовое представление молекул

1. Узлы представляют собой молекулярные атомы. Каждый узел имеет свой уникальный идентификатор (например, атомный номер) и может содержать дополнительные атрибуты (например, химические свойства атома).

2. Рёбра соединяют узлы и представляют химические связи между атомами. В графовых представлениях молекул обычно используются два типа рёбер:

   - `edge_index`: Это список пар узлов, которые соединены рёбрами. Например, если у нас есть ребро между атомами 1 и 2, то (1, 2) будет присутствовать в `edge_index`. Таким образом, размер этого поля $2 * |E|$, где $|E|$ – количество рёбер в графе.
   - `edge_attr`: Это матрица признаков рёбер, таких как тип связи (одинарная, двойная, тройная) или расстояние между атомами.

3. Атрибуты узлов (`node_attr`): Это матрица, состоящая их дополнительных характеристик каждого узла. Например, масса атома, заряд, гибридизация и т. д.

4. Количество узлов (`num_nodes`): Это общее количество атомов в молекуле.

В графовых моделях, таких как MolCLR и Graphormer, на вход графы подаются в формате, который содержит данные поля, поэтому они играют важную роль в обучении подобных моделей.

3.9 Библиотека RDKit

RDKit [22] – это библиотека для хемоинформатики и машинного обучения, написанная на C++ и Python. RDKit предоставляет множество функций для работы с молекулами, включая чтение, отображение, запись и многие другие операции. Есть возможность создавать молекулы из SMILES-кодов, файлов формата MOL и блоков

данных MOL, а также поддерживается отображение молекул в виде графических изображений.

Пример получения обьекта молекулы из SMILES:

```
rdkit.Chem.MolFromSmiles('Cc1ccccc1')
```

3.10 Датасет ChemBL

ChEMBL [23] – это база данных биоактивных молекул с лекарственными свойствами. Она объединяет химические, биоактивные и геномные данные для поддержки перевода геномной информации в эффективные новые лекарства.

На основе данных из данного датасета можно получить молекулярные фингерпринты или SMILES и обучить модель машинного обучения для регрессии молекулярных свойств, используя различные алгоритмы, такие как графовые нейронные сети (GNN), регрессию или другие методы. Таккже можно использовать датасет ChEMBL для тестирования полученных моделей, так как он предоставляет информацию о биоактивных молекулах и их свойствах.

3.11 Оптимизатор AdamW

AdamW [24] — один из самых эффективных алгоритмов оптимизации для обучения нейронных сетей. Это модификация оптимизационного алгоритма Adam (adaptive moment estimation), который сочетает в себе идеи RMSProp и оптимизатора импульса. Adam использует адаптивную оценку моментов первого и второго порядка.

- Основная идея AdamW заключается в добавлении метода декремента весов (weight decay) к Adam.
- Декремент весов помогает бороться с переобучением, регуляризуя модель.
- AdamW более устойчив к большим значениям весов и улучшает обобщающую способность модели.

3.12 Функции потерь

3.12.1 L1-Loss

Для L1-потерь обычно используется MAE (Mean Absolute Error):

$$L1 = \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{4}$$

где N - количество элементов во входе y, y_i - фактическое значение для i-го примера, а $\hat{y}_i$ - предсказанное значение для i-го примера.

3.12.2 MSE-Loss

Функция потерь Mean Squared Error (MSE) измеряет среднеквадратичную ошибку (квадрат нормы L2) между каждым элементом входа x и целевым значением y. Она описывается следующим образом:

$$\ell(y, \hat{y}) = \frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y}_i)^2 \tag{5}$$

где N - количество элементов во входе y, y_i - фактическое значение для i-го примера, а $\hat{y}_i$ - предсказанное значение для i-го примера.

3.12.3 CosineSimilarityLoss

Cosine Similarity Loss используется для измерения сходства между двумя векторами на основе косинусного сходства. Это часто применяется для оценки степени схожести между векторами. Значение косинусного сходства находится в диапазоне от -1 до 1, где 0 указывает на ортогональность, а значения, близкие к 1 по модулю, указывают на большее сходство. Формула косинусного расстояния:

$$sim(x_1, x_2) = \frac{x_1 * x_2}{max(\|x_1\| * \|x_2\|, \epsilon)} \tag{6}$$

3.12.4 NTXentLoss

NT-XentLoss (Normalized Temperature-scaled Cross Entropy Loss) — это функция потерь, используемая в глубоком обучении. Она измеряет косинусное сходство между двумя векторами и использует параметр температуры для балансировки положительных и отрицательных пар. Пусть $sim(u, v)$ обозначает косинусное сходство между векторами u и v. Тогда функция потерь для положительной пары примеров (i, j) выглядит следующим образом:

$$l_{i,j} = -\log\left(\frac{e^{sim(u_i, v_j)/\tau}}{\sum_{k=1}^{N} e^{sim(u_i, v_k)/\tau}}\right) \tag{7}$$

где $sim(u_i, v_j)$ - косинусное сходство между векторами u_i и v_j, N - общее количество примеров, τ (температура) - параметр, который регулирует вклад положительных и отрицательных пар.

Эта функция потерь позволяет обучать эмбеддинги так, чтобы они были близки для положительных пар и далеки для отрицательных пар, что полезно для задач, связаных с поиском сходства между векторами, например, в задачах ранжирования или рекомендации.

4 Исследование и построение решения задачи

4.1 Декомпозиция задачи

Для решения поставленной задачи необходимо разбить её на ряд подзадач, каждая из которых должна быть достаточно простой для непосредственного решения. Процесс декомпозиции включает в себя следующие шаги:

1. **Сбор требований и анализ проблемы:**
   - Определение целей исследования и задач, которые необходимо решить.
   - Исследование существующих решений и определение их ограничений.
2. **Исследование и выбор подходов:**
   - Анализ различных методов и технологий, применимых для решения задачи.
   - Оценка плюсов и минусов различных моделей и подходов на основе научных публикаций и предыдуших исследований.
   - Выбор оптимального подхода для каждой из подзадач.
3. **Проектирование архитектуры системы:**
   - Разработка общей архитектуры решения, включающей интеграцию всех компонентов.
   - Определение взаимодействий между различными частями системы (например, как данные будут передаваться между моделями RoBERTa и GNN).
4. **Подготовка данных:**
   - Сбор исходных данных.
   - Очистка данных от выбросов и слишком больших молекул.
   - Нормализация данныхю
   - Преобразование данных в необходимые форматы (ECFP и графовое представление).
   - Предобработка данных для Graphormer (добавление необходимых полей, обьединение графов в батчи).
5. **Разработка и настройка моделей:**
   - Разработка и настройка токенайзера для модели RoBERTa.
   - Определение и настройка архитектур для модели GNN.
   - Настройка модели Graphormer.
   - Подбор гиперпараметров для моделей.
   - Выбор наиболее подходящей модели для задачи.
   - Разработка baseline-модели, которая объединяет в себе два подхода.
6. **Обучение и тестирование моделей:**
   - Обучение модели RoBERTa с использованием Masked Language Modeling.
   - Обучение моделей GNN: MolCLR и Graphormer.

- Тестирование обученных моделей на валидационном наборе данных.

7. **Интеграция и согласование эмбеддингов:**
   - Проекция эмбеддингов из различных моделей в общее пространство.
   - Оптимизация сходства эмбеддингов с использованием метрик, таких как косинусное расстояние.
8. **Анализ и интерпретация результатов:**
   - Проведение экспериментов для оценки качества полученных эмбеддингов.
   - Интерпретация результатов и формулирование выводов.

5 Описание практической части

В это разделе описываются все проведенные эксперименты, а также их результаты. Важно отметить, что все вставляемые фрагменты кода используют библиотеки PyTorch [25], RDKit [22] и HuggingFace [26], а также предполагают наличие определенных глобальных переменных и конфигураций, которые не указаны в коде.

5.1 Препроцессинг данных

Препроцессинг данных начинается с использования библиотеки RDKit [22] для преобразования строковых представлений молекул в формате SMILES в структурированные графовые объекты. Это позволяет перевести химическую информацию, содержащуюся в SMILES, в формат, который может быть эффективно обработан современными графовыми нейронными сетями.

1. Импорт необходимых модулей из библиотеки RDKit, определение списков атомов, хиральности и типов связей (были взяты стандартные списки из RDKit).

2. Создание функции `get_graph_columns`, которая принимает SMILES-строку и возвращает графовое представление молекулы (матрицу связей, признаки узлов, рёбер, а также количество узлов в графе).

3. Применение функций RDKit для преобразования SMILES в объект молекулы, из которого затем извлекается информация об атомах и связях для создания графового представления.

4. Построения матрицы признаков узлов графа (для чего используются кодирование атомных номеров, хиральностей и типов связей).

5. Применение функции `get_graph_columns` к столбцу датасета, содержащему SMILES.

Этот процесс позволяет преобразовать большое количество молекул, представленных в виде SMILES, в структурированные графовые данные для предсказания молекулярных свойств с помощью GNN.

Данный препроцессинг производится над датасетом ChemBL [23], где содержатся также необходимые отпечатки молекул, которые будут использоваться как вход для трансформера.

5.2 Обучение RoBERTa

Обучение модели RoBERTa для задач хемоинформатики требует специальной настройки и адаптации стандартных процедур обучения трансформеров. В этом разделе описывается процесс настройки токенайзера, методика обучения с маскированием и тестирование обученной модели, а также анализ полученных результатов.

5.2.1 Настройка токенайзера

Используя библиотеку HuggingFace [26], был создан ByteLevelBPETokenizer. Процесс настройки включает в себя следующие этапы:

1. Обучение токенайзера на полученных файлах с целью создания словаря токенов и правил их слияния. Параметры обучения включают размер словаря, минимальную частоту токенов и специальные токены ($< s >, < pad >, < /s >, < unk >, < mask >$).

2. Сохранение обученного токенайзера, включая файлы `vocab.json` и `merges.txt`, которые содержат информацию о словаре токенов и правилах слияния.

3. Тестирование токенайзера на примере молекулярной последовательности для проверки корректности токенизации и присвоения идентификаторов токенам.

Токенайзер позволяет уменьшить размер словаря. Это позволяет снизить размерность пространства наших значений хеш-функции (используемой в алгоритме ECFP) с 2^{32} до размерности пространства токенов (примерно 10000).

5.2.2 Обучение маскированием

Методика обучения с маскированием (Masked Language Modeling, MLM) является ключевой для обучения модели RoBERTa. Этот подход позволяет модели учиться предсказывать исходные значения замаскированных токенов, что способствует лучшему пониманию контекста и структуры молекул. В качестве основы была взята модель `RobertaForMaskedLM` из библиотеки HuggingFace [26]. Процесс обучения включает следующие шаги:

1. Использование функции маскирования `mlm`, которая создает случайную маску для 15% токенов в тензоре, исключая специальные токены, такие как начало (<s>), конец последовательности (</s>) и паддинг(<pad>).

2. Применение функции `tokenize_ecfps` для токенизации и маскирования ECFP последовательностей, подготовки входных данных, масок внимания и меток для обучения.

3. Создание загрузчиков данных (DataLoaders) для обучающего, валидационного и тестового наборов данных, которые обеспечивают эффективную подачу данных в модель во время обучения.

4. Инициализация модели `RobertaForMaskedLM` с использованием определенной конфигурации `RobertaConfig`, где задаются параметры модели (стандартный размер словаря - 30522, максимальная длина позиционных эмбеддингов - 512 + 2, стандартный для трансформера размер скрытых слоев - 768 и количество скрытых слоев - 6). Модель обучается без загрузки весов, from scratch.

5. Инициализация оптимизатора AdamW [24] с шагом обучения 1e-5.

6. Применение цикла обучения, включающего в себя подсчет потерь, обратное распространение ошибки, шаг оптимизации и обнуление градиентов.

7. Логирование функции потерь и метрик, таких как accurracy, precision recall, F1-score, с помощью инструмента `wandb` [27] для отслеживания прогресса обучения.

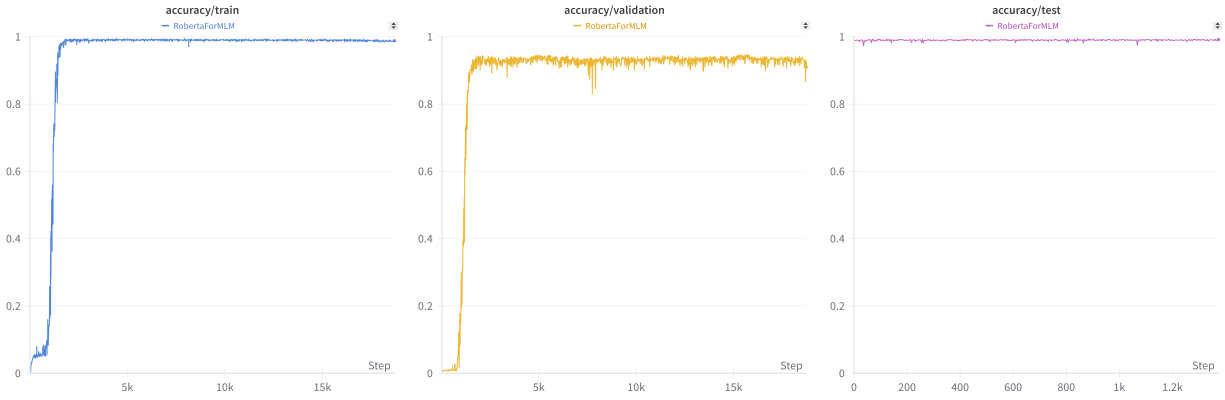
8. Проведение валидации модели на валидационном наборе данных для оценки ее способности к обобщению и предсказанию.

5.2.3 Тестирование обученной модели и результаты

После завершения процесса обучения результаты модели RoBERTa были протестированы на независимом (тестовом) наборе данных. Использовались следующие метрики качества из библиотеки sklearn [28] (метки были сагрегированы как взвешенное среднее):

- `accuracy_score`: 0.86 (отражает долю правильно классифицированных примеров)
- `precision_score`: 0.89 (показывает долю релевантных экземпляров среди всех экземпляров, выбранных моделью)
- `recall_score`: 0.86 (отражает способность модели обнаруживать все релевантные случаи в данных)
- `f1_score`: 0.87 (представляет собой сбалансированную меру precision и recall)

Графики точности для обучения (a), валидации (b) и тестирования (c) представлены на Рисунке 4. Эти графики иллюстрируют динамику изменения точности модели на различных этапах её оценки и позволяют визуально оценить уровень обобщающей способности модели и её стабильность в различных условиях.



(a) **(b)** **(c)**

Рис. 4: Графики метрики accuracy: (a) train, (b) validation, (c) test

Результаты тестирования подтверждают высокую производительность модели RoBERTa в задаче предсказания молекулярных свойств, что делает её перпективным инструментом для дальнейших исследований в области хемоинформатики.

5.3 Обучение MolCLR

Процесс обучения включает следующие шаги:

1. В начале исследования был создан экземпляр `MoleculeDataset` на основе датасета ChemBL [23]. Далее были определены индексы для обучающего, валидационного и тестового наборов данных. Для каждого из этих наборов данных был создан `DataLoader` из библиотеки `torch_geometric` [29], который использовался для итерации по данным в процессе обучения и валидации.

2. В качестве основы для обучения была выбрана самописная модель `MolecularGraph`. Эта модель включает в себя графовую модель (в зависимости от конфигурации это может быть `GINet` или `GCN`), а также линейный слой для предсказания. GNN была инициализирована с предварительно обученными весами.

3. В ходе обучения вычислялась функция потерь между предсказанными и истинными метками и выполнялось обратное распространение ошибки. В качестве метрики была выбрана L1-норма. Результатом одной эпохи было усреднение loss-функции по всей эпохе.

Код цикла обучения:

```
model = MolecularGraph().to(device)
loss_func = torch.nn.L1Loss()
def train_loop():
    train_tqdm = tqdm(train_dataloader, unit="batch")
    train_tqdm.set_description(f'Epoch {epoch_counter}')
    loss_sum = 0

    model.train()
    for (batch, labels_batch) in train_tqdm:
        optimizer.zero_grad()

        graph_batch = batch.to(device)
        labels_batch = labels_batch.to(device)

        predicted_labels = model(graph_batch)

        loss = loss_func(predicted_labels, labels_batch)
        loss.backward()

        loss_sum += loss.item()

        optimizer.step()
        train_tqdm.set_postfix(loss=loss.item())
    return loss_sum / len(train_dataloader)
```

4. В цикле валидации `eval_loop()` был выполнен проход по каждому батчу в валидационном `eval_dataloader`. Процесс был аналогичен циклу обучения, однако в этом случае веса модели не обновлялись. Валидация происходила каждую эпоху.

5. После каждой эпохи проводилось сравнение фукнции потерь на обучающем и валидационном наборах данных. Это позволило отслеживать процесс обучения и валидации модели в динамике.

Данное обучение было проведено на 4 физических свойствах молекул: Polar Surface Area, AlogP, Bioactivities и Molecular Weight. В ходе эксперимента были получены следующие результаты. На рисунке 5 представлены графики функции потерь в процессе обучения (a) и валидации (b) для этих четырех свойств.

На тестовой выборке были получены следующие значения функции потерь:

- Molecular Weight: 99.521
- Polar Surface Area: 20.127
- AlogP: 1.019
- Bioactivities: 5.74

В данном датасете свойство Molecular Weight лежит в пределах [77.06; 1060.97] со средним 388.6, свойство Polar Surface Area лежит в пределах [0.0; 392.73] со средним 78.5. свойство AlogP лежит в пределах [-5.61; 11.97] со средним 3.33, свойство Bioactivities лежит в пределах [1.0; 3247.0] со средним 12.1. Эти данные показывают, что диапазоны и средние значения для каждого из четырех свойств существенно различаются. Поэтому, чтобы сравнить производительность модели на разных свойствах, необходимо рассмотреть относительные значения функции потерь:

- Molecular Weight: 99.521/388.6 = 0.26
- Polar Surface Area: 20.127/78.5 = 0.26
- AlogP: 1.019/3.33 = 0.31
- Bioactivities: 5.74/12.1 = 0.47

Эти результаты указывают на то, что модель наиболее точно предсказывает свойства Molecular Weight и Polar Surface Area, в то время как для свойства Bioactivities значение функции потерь значительно выше, что может указывать на необходимость дальнейшей оптимизации модели для этого свойства.

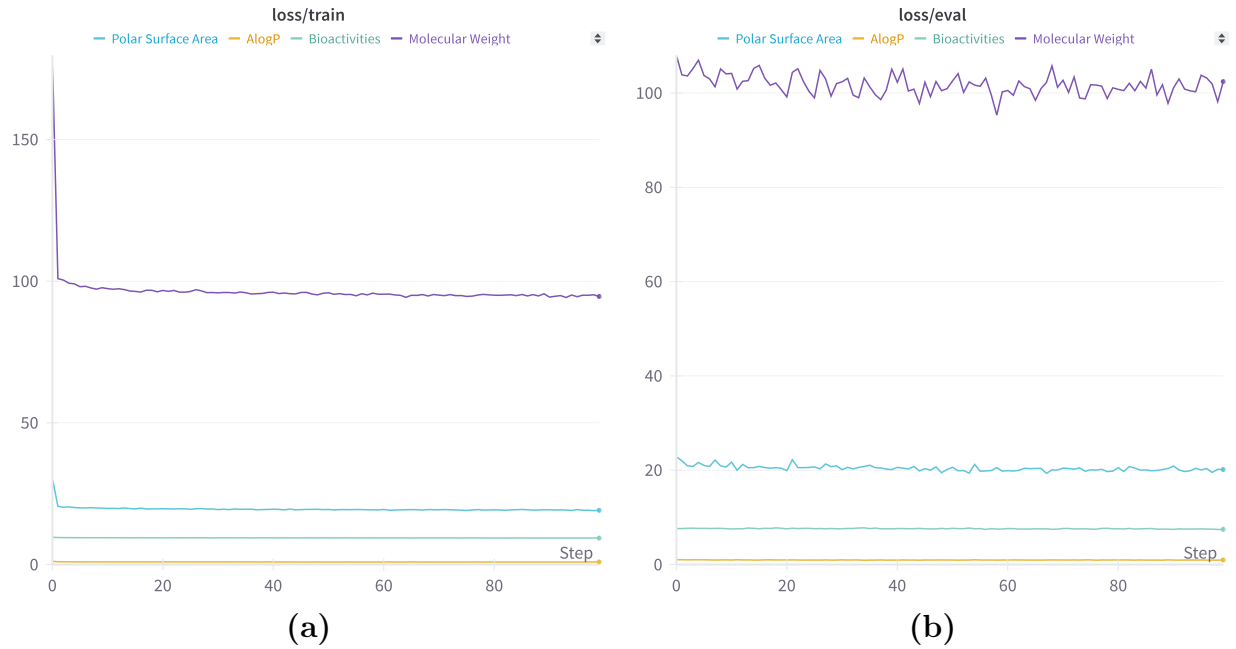


Рис. 5: Графики функции потерь L1-loss: (a) train, (b) validation

5.4 Обучение Graphormer

Обучение Graphormer включает в себя использование специализированных слоёв внимания, которые позволяют модели учитывать глобальные структурные особенности графа. Этап обучения направлен на минимизацию функции потерь и оптимизацию параметров модели для достижения наилучшей производительности. В данном эксперименте используется функция потерь MSE (Mean Squared Error). Для оптимизации параметров применяется классический оптимизатор AdamW с коэффициентом обучения равным 5e-5. Кроме того, используется линейный планировщик скорости обучения.

1. Сначала из части датасета ChemBL [23] инстанцируется экземпляр класса `Dataset` библиотеки HuggingFace [26]. Далее идет разделение всего набора данных на обучающий, валидационный и тестовый наборы. Это дает 80% данных для обучения, 10% для валидации и 10% для тестирования.

2. Затем применяется специальная функция `preprocess_item` к каждому элементу в наборах данных. Это преобразует каждый элемент в формат, подходящий для модели Graphormer (добавляются еще 7 тензоров для каждой молекулы).

3. Инициализируется модель `GraphormerForGraphClassification` (использовался чекпоинт `clefourrier/graphormer-base-pcqm4mv1`). Также устанавливается параметр `num_classes` равным 1, поскольку предсказывается одно свойство.

4. Инициализируется GraphormerDataCollator, который позволяет обьединять данные графов в батчи (в отличие от обычных GNN, где `torch_geometric.data` обьекты графов подаются на вход по одному). Данный обраточик данных подается в инициализацию Dataloaders для каждого набора данных.

5. Инициализируется **оптимизаторы обучения:**

```
from transformers import AdamW, get_scheduler

num_epoch = 100
num_training_steps = num_epoch * len(train_dataloader)

optimizer = AdamW(model.parameters(), lr=5e-5)
lr_scheduler = get_scheduler(
    'linear',
    optimizer = optimizer,
    num_warmup_steps = 5,
    num_training_steps = num_training_steps,
)
```

6. Далее исполняется стандартный **цикл обучения:**

```
progress_bar_train = tqdm(range(num_training_steps))

model.train()
train_epoch_loss = 0
for batch in train_dataloader:
    input_batch = { k: v.to(device) for k, v in batch.items() }

    outputs = model(**input_batch)

    loss = outputs["loss"]
    loss.backward()
    train_epoch_loss += loss.item()

    optimizer.step()
    lr_scheduler.step()
    optimizer.zero_grad()
    progress_bar_train.update(1)
```

7. В цикле валидации итерация проходт по каждому батчу в `eval_dataloader`. Процесс похож на цикл обучения, но здесь не обновляются веса модели.

8. После каждой эпохи сравниваются функции потерь на обучающем и валидационном наборах данных. Эти данные сохраняются с помощью библиотеки wandb [27]. С помощью одноименного веб-сервиса можно будет отследить, как модель обучалась и валидировалась в процессе обучения.

Результаты работы данного алгоритма показаны на рисунке 6. На рисунке (a) изображен график обучения Graphormer с помощью регресии молекулярного свойсва

Molecular Weight. Также здесь представлен график обучения модели MolCLR для сравнения результатов. Рядом, на рисунке (b), показаны результаты валидации этих двух моделей. Исходя из этих данных, понятно, что трансформер Graphormer имеет потенциал предсказывания молекулярных свойств выше, чем модель MolCLR, основанная на классической модели свертки графов. Поэтому исследуемый Graphormer может быть использован как альтернатива или дополнение к модели MolCLR.

В ходе исследования была предпринята попытка нормализации данных с использованием MinMaxScaler из библиотеки sklearn [28]. Однако, после нормализации данные оказывались слишком близкими к нулю, что приводило к погрешностям в вычислениях и делало графики менее информативными. В связи с этим, результаты исследования показаны с использованием исходных, ненормализованных данных для обучения модели Graphormer.

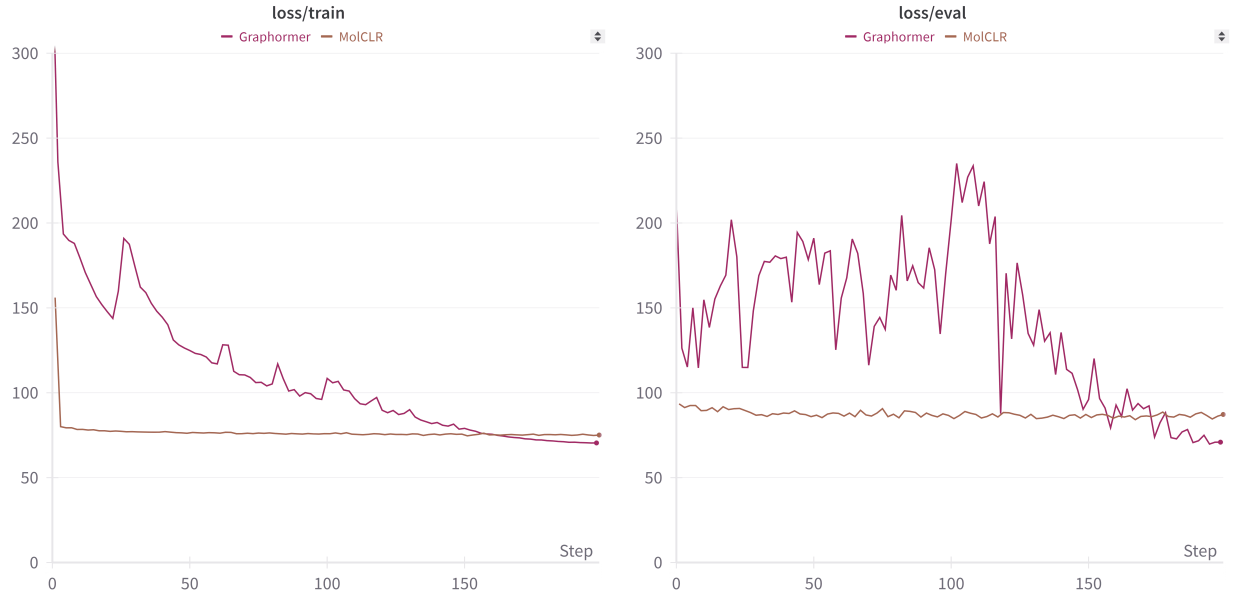


Рис. 6: Сравнение графиков функций потерь Graphormer и MolCLR: (a) train, (b) validation

5.5 Разработка и реализация baseline-модели

В данном разделе будет рассмотрен подход к обучению двух моделей на основе разных представлений молекулярных данных. В качестве отправной точки, или “бейзлайна”, были выбраны две модели: RoBERTa [11] и MolCLR [3]. Эти модели были выбраны из-за их эффективности в обработке текстовых и графовых данных соответственно, а также из-за относительной простоты их объединения. Для приближения эмбеддингов, полученных от каждой из этих моделей, использовалась функция косинусоидальной близости CosineSimilarityLoss. Обучение проводилось на датасете ChemBL. [23] Общая схема модели изображена на рисунке 7.

5.5.1 Описание алгоритма обучения

1. Инициализация класса MoleculeDataset, который наследуется от класса Dataset из библиотеки `torch_geometric` и предназначен для создания набора данных из молекулярных структур, представленных в формате SMILES. Полученный набор данных состоит из токенов для входа модели трансформера, а также двух графовых представлений одной молекулы, которое используется в MoLCLR. Токены

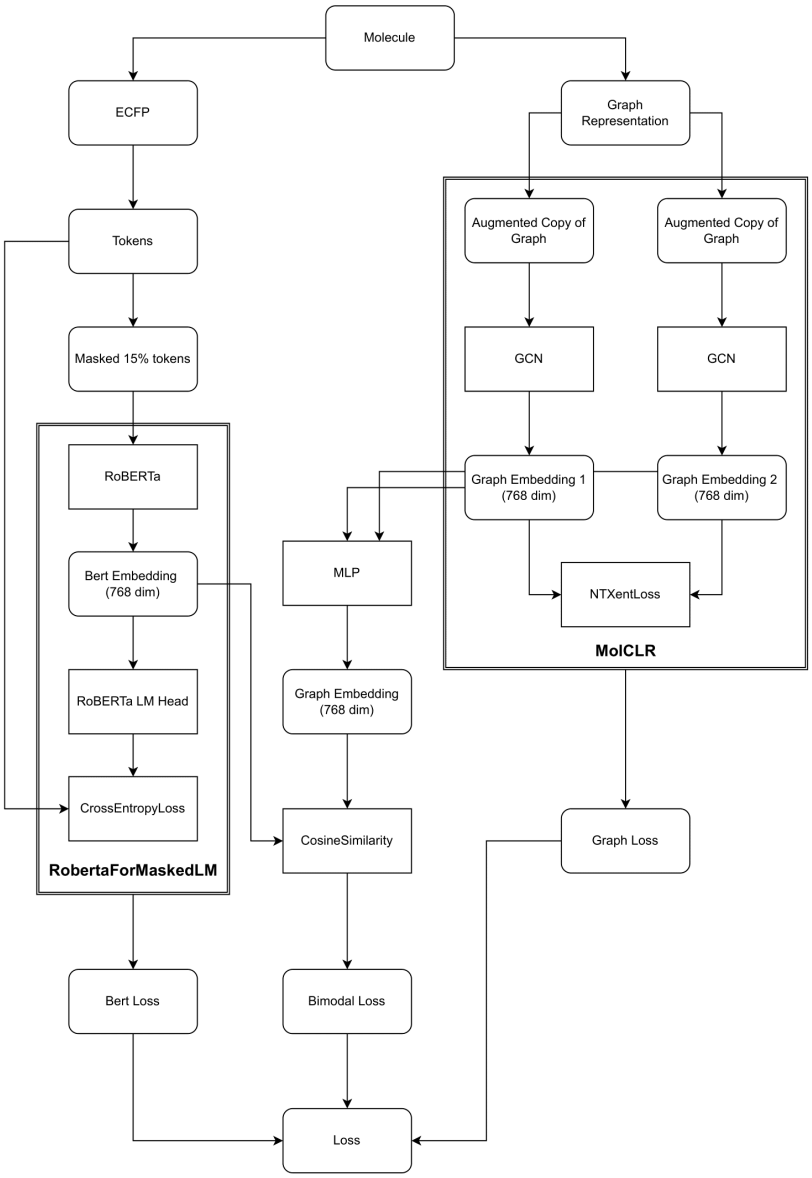


Рис. 7: Архитектура объединения двух моделей RoBERTa и MolCLR.

получаются применением вышеопределенных функций `tokenize` и `mlm`. Графовые представления являются агментированными копиями графа молекулы. Метод класса MoleculeDataset `__getitem__`:

```
def __getitem__(self, index):
    node_feat, edge_index, edge_attr, num_nodes, num_edges = self.
        get_graph_from_smiles(self.dataset['Smiles'][index])

    graph_i = self.get_augmented_graph_copy(node_feat, edge_index, edge_attr,
        num_nodes, num_edges)
    graph_j = self.get_augmented_graph_copy(node_feat, edge_index, edge_attr,
        num_nodes, num_edges)

    ecfp = self.dataset['ecfp1'][index]
    data_for_bert = self.apply_mlm(self.tokenize(ecfp))
    return data_for_bert, graph_i, graph_j
```

2. Инициализация класса MolecularBertGraph: Этот класс представляет собой модель, которая объединяет модель `RobertaForMaskedLM` и графовую нейронную сеть MolCLR:

```
from transformers import RobertaForMaskedLM
from transformers import RobertaConfig

if config['model_type'] == 'gin':
    from MolCLR.models.ginet_molclr import GINet as GraphModel
elif config['model_type'] == 'gcn':
    from MolCLR.models.gcn_molclr import GCN as GraphModel
else:
    raise ValueError('GNN model is not defined in config.')
from MolCLR.utils.nt_xent import NTXentLoss

class MolecularBertGraph(torch.nn.Module):
    def __init__(self):
        super(MolecularBertGraph, self).__init__()
        self.bert = RobertaForMaskedLM(roberta_config)
        self.graph_model = GraphModel(**config['model'])
        self.out_graph_linear = torch.nn.Linear(768 * 2, 768, bias=True)
        # contrastive loss for MolCLR
        self.nt_xent_criterion = NTXentLoss(self.batch_size, **config['loss'])
        # cosine distance as loss between models
        self.cosine_sim = torch.nn.CosineSimilarity(dim=-1)

    def forward(self, bert_batch, graph_batch1, graph_batch2):
        bert_output = self.bert(bert_batch)
        bert_loss = bert_output.loss
        bert_emb = bert_output.hidden_states[0][:, 0, :] # CLS embedding

        graph_loss, hidden_states_1, hidden_states_2 = self.graph_step(
            graph_batch1, graph_batch2)
        graph_emb = self.out_graph_linear(torch.cat((hidden_states_1,
            hidden_states_2), dim=-1))

        bimodal_loss = ((1 - self.cosine_sim(bert_emb, graph_emb))**2).mean()
        return bert_loss, graph_loss, bimodal_loss

    def graph_step(self, xis, xjs):
        ris, zis = self.graph_model(xis)
        rjs, zjs = self.graph_model(xjs)
        # normalize projection feature vectors
        zis = torch.nn.functional.normalize(zis, dim=1)
        zjs = torch.nn.functional.normalize(zjs, dim=1)

        loss = self.nt_xent_criterion(zis, zjs)
        return loss, ris, rjs
```

3. Создание экземпляра MoleculeDataset, загрузчиков данных для обучающего и валидационного наборов данных, а также экземпляра MolecularBertGraph.

4. Инициализация стандартного оптимизиматора AdamW с lr равным 5e-3.

5. В цикле обучения вычисляется значения функции потерь `bert_loss`, `graph_loss`, `bimodal_loss` на каждом батче из `train_dataloader`. Финальное значение функции потерь вычисляется по формуле

$$\texttt{loss} = \alpha * \texttt{bert\_loss} + \beta * \texttt{graph\_loss} + \gamma * \texttt{bimodal\_loss}$$

   где α, β, γ - соответсвующие коэффициенты (гиперпараметры). Далее происходит обратное распространение ошибки. Результатом одной эпохи было усреднение всех значений функции потерь по всей эпохе.

6. В цикле валидации `eval_loop()` был выполнен проход по каждому батчу в валидационном `eval_dataloader`. Процесс был аналогичен циклу обучения, однако в этом случае веса модели не обновлялись. Валидация происходила каждую эпоху.

7. Далее идет основной цикл обучения, который выполняет заданное количество эпох обучения. В каждой эпохе выполняются обучающий и валидационный циклы, а также обновляются логи и сохраняются веса модели.

В следующем разделе представлены графики обучения и валидации для всех четырех функций потерь: `bert_loss`, `graph_loss`, `bimodal_loss` и `loss`. Эти графики помогут лучше понять динамику обучения и валидации модели, а также влияние каждой из компонент потери на общую функцию потерь.

5.5.2 Результаты baseline-эксперимента

Baseline эксперимент был проведен с двумя вариантами графовых моделей: GCN и GIN. На рисунке 8 представлен график динамики общей функции потерь для двух моделей: RobertaForMaskedLM + MolCLR (GIN) и RobertaForMaskedLM + MolCLR (GCN) в процессе обучения (a) и в процессе валидации (b). Ось абсцисс представляет собой количество эпох обучения, а ось ординат - значения функции потерь, варьирующиеся от 0 до 1.

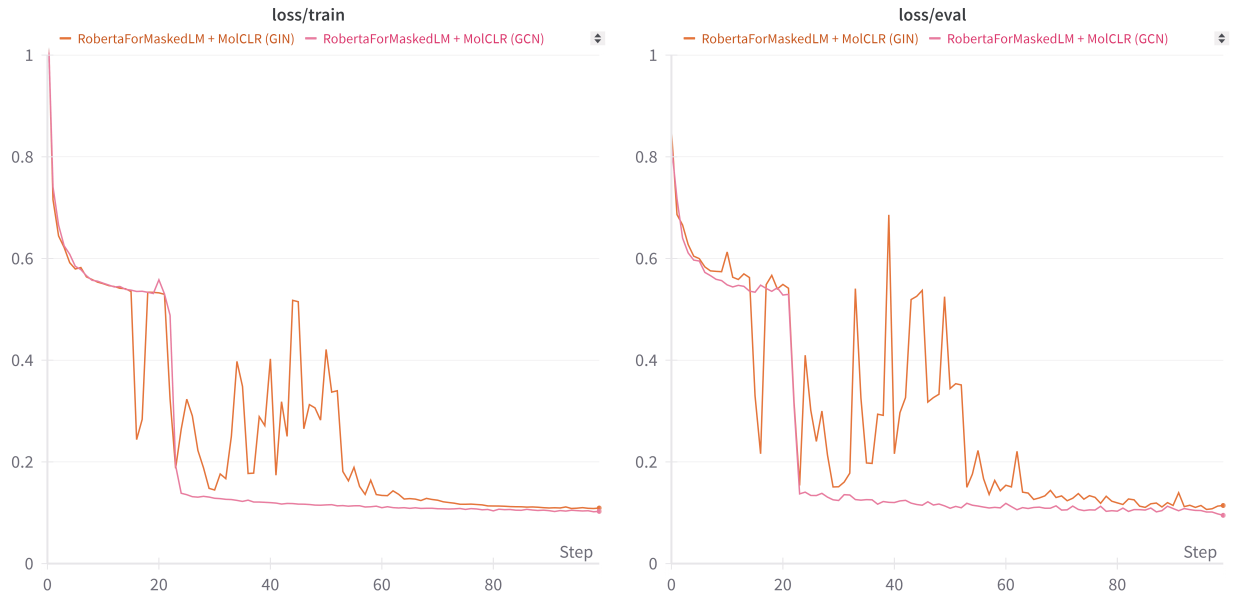


Рис. 8: Графики общей функций потерь в baseline-модели: (a) train, (b) validation

Обе кривые на графике (a) рисунка 8 демонстрируют типичное поведение функции потерь в процессе обучения: быстрое уменьшение в начале, когда происходит основное обучение, за которым следуют более мелкие колебания и в конечном итоге стабилизация, по мере сходимости модели к оптимальным параметрам.

Красная линия, соотвествующая модели с графовой GIN, показывает резкое снижение потерь с начального значения в течение первых нескольких эпох, после чего следуют колебания с уменьшающейся амплитудой по мере продвижения вдоль шагов, в конечном итоге выравниваясь.

Оранжевая линия, соотвествующая модели с графовой GCN, также показывает начальное резкое снижение значений потерь, но с меньшими колебаниями по сравнению с красной линией, и далее она уменьшается более стабильно, приближаясь к своей асимптоте.

График валидации (b) демонстрирует аналогичные изменения для обоих конфигураций модели, что характеризует успешное проведение эксперимента и качественное обучение полученной модели.

Далее, на рисунках 9, 10 и 11 представлены графики функций потерь RoBERTa, MolCLR и бимодальной функции потерь в baseline-модели соответсвенно. Графики показаны при обучении модели (a) и при валидации (b), а также для двух вариантов графовой модели GCN и GIN. График междумодельной loss-функции отнормирован для наглядности.

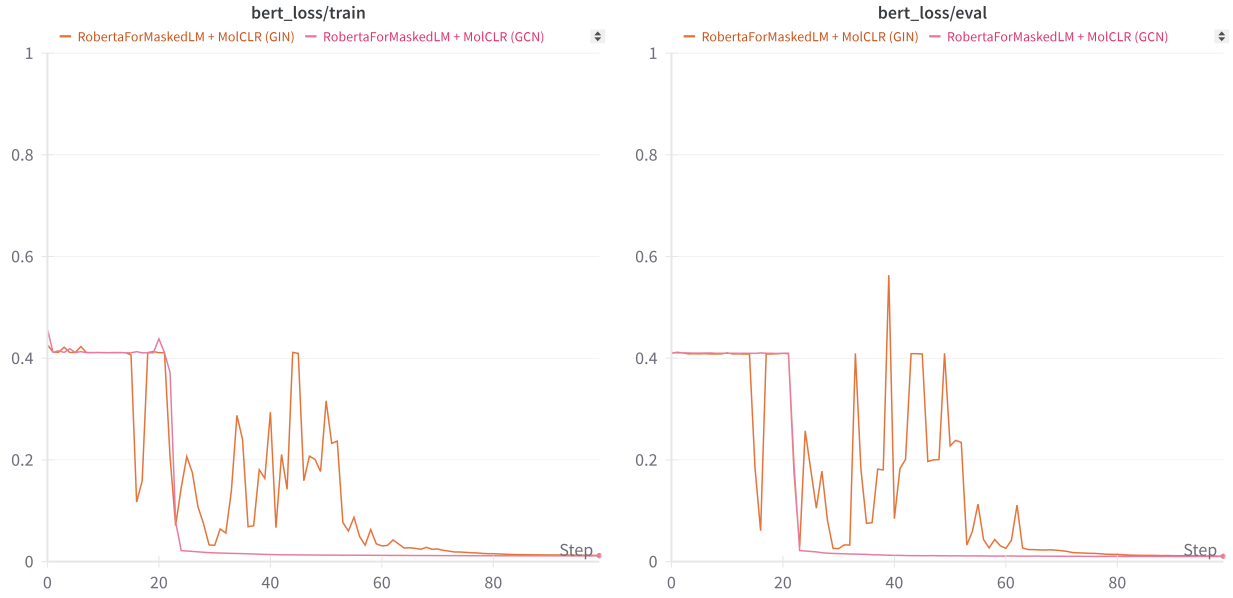
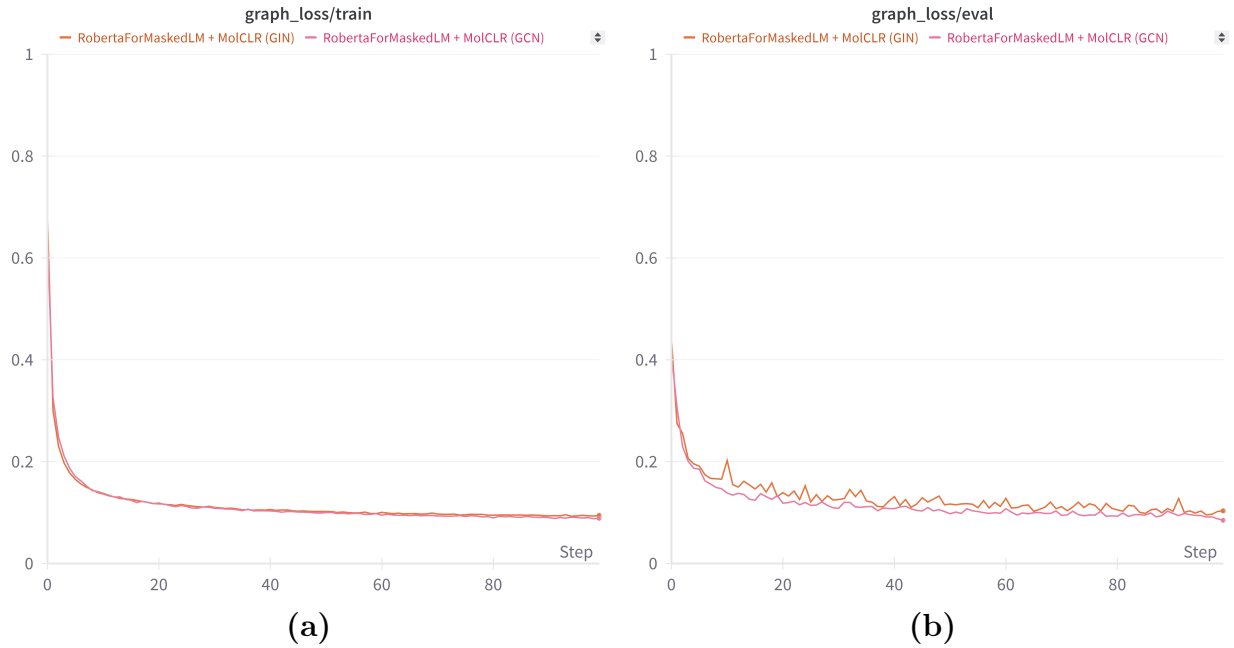


Рис. 9: Графики функций потерь RoBERTa в baseline-модели: (a) train, (b) validation

Как видно на графике 9, функция потерь RoBERTa играет основную роль в общей функции потерь, поскольку она отражает эффективность модели RoBERTa в предсказании маскированных токенов в молекулярных отпечатках ECFP. Оптимизация этой функции потерь ведет к улучшению качества эмбеддингов, получаемых от модели RoBERTa, что, в свою очередь, влияет на общую функцию потерь и эффективность всей модели.

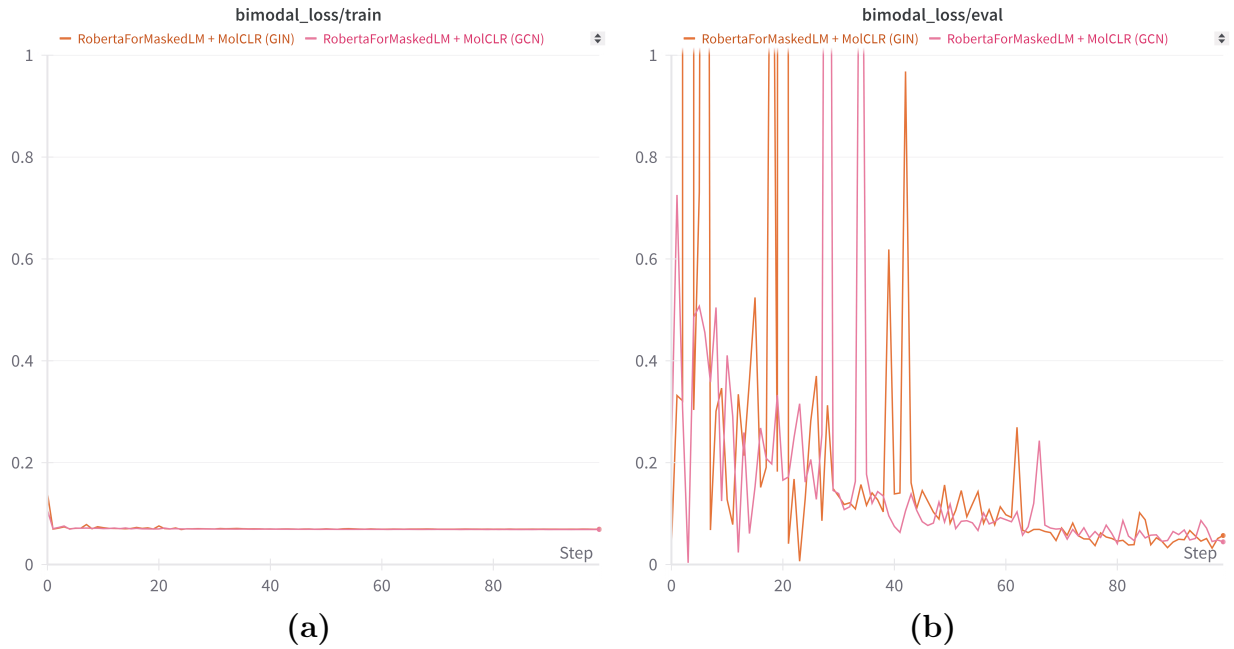
На данных рисунках видны резкие измения графика, которые могли быть вызваны некорректно подобранным шагом обучения. Эти колебания характеризуются резкими всплесками и спадами, что отражает итеративный процесс оптимизации в процессе обучения. GCN, демонстрирует схожую динамику, но с меньшей степенью волатильности.



(a) (b)

Рис. 10: Графики функций потерь MolCLR в baseline-модели: (a) train, (b) validation

Функция потерь MolCLR также играет важную роль в общей функции потерь, поскольку она отражает эффективность модели MolCLR в генерации эмбеддингов графов молекул. Это подтверждается графиком 10, где на обучении и валидации обе модели демонстрируют схожую динамику уменьшения функции потерь. Это указывает на то, что обе модели успешно обучаются, улучшают свою производительность с течением времени, однако быстро достигают "плато".



(a) (b)

Рис. 11: Графики бимодальной функций потерь в baseline-модели: (a) train, (b) validation

На представленном выше графике рисунка 11 отображены динамики бимодальной функции потерь на этапе обучени и валидации. Данная функция является косинусоидальным расстоянием между внутренними представлениями RoBERTa и MolCLR. Начальное значение (расстояние между случайными эмбеддингами) равно приблизитель-

но 1, однако оно быстро сходится к числу, сильно меньше 0.1. Поэтому стоит отметить, что значения функции потерь на этом графике умножены на 100 для наглядности, поскольку исходные значения `bimodal_loss` получаются слишком близкими к нулю.

Несмотря на то, что функция потерь между двумя моделями является важной составляющей общей функции потерь, ее вклад в общую функцию потерь оказывается относительно небольшим. Полученные результаты свидетельствует о высокой степени согласованности эмбеддингов, получаемых от моделей RoBERTa и MolCLR, а также достаточно быстрой сходимости данной функции похожести эмбеддингов при обучении.

Про коэффиценты общего loss: В ходе экспериментов была предпринята попытка установить коэффициенты α, β, γ обратно пропорциональными начальным значениям соответствующих функций потерь. Однако, как показали результаты, такой подход приводит к эффекту, аналогичному случаю, когда все коэффициенты равны 1. Это может быть связано с тем, что обратная пропорциональность начальных значений потерь приводит к нормализации вклада каждой компоненты потери в общую потерю. В результате, каждая компонента потери вносит примерно одинаковый вклад в общую потерю.

Таким образом, для дальнейшего анализа были выбраны результаты, полученные при $\alpha = \beta = \gamma = 1.0$. В этом случае все функции потерь вносят одинаковый вклад в общую функцию потерь.

6 Заключение

В ходе выполнения данной дипломной работы была успешно решена задача исследования и применения методов обучения и fine-tuning для моделей машинного обучения молекулярных структур. Были достигнуты следующие ключевые результаты:

1. Исследование уже существующих подходов к классификации и регрессии свойств молекулярных структур. Выбор наиболее успешных подходов, для последующей их проверки и применения (реализации) в решении задачи.

2. Обучение и fine-tuning существующих моделей. Проведено обучение, валидация и тестирование моделей RoBERTa, Graphormer и MolCLR с использованием различных подходов к инициализации и настройке параметров.

3. Разработка и обучение новой модели. Обьединение существующие моделей RoBERTa и MolCLR, их совместное обучение и валидация. Результаты показали высокую степень согласованности эмбеддингов, получаемых от обеих моделей, а также высокое качество обучения данных моделей.

4. Исследование новых подходов к классификации и регрессии. В процессе работы были изучены и протестированы новые методы классификации и регрессии молекулярных свойств. Применение трансформеров и графовых нейронных сетей позволило значительно повысить точность предсказаний по сравнению с традиционными методами.

Настоящая работа вносит вклад в развитие методов машинного обучения для анализа и предсказания свойств химических соединений, открывая новые возможности для их практического применения.

Список литературы

[1] *Walid Ahmad, Elana Simon et al.* ChemBERTa-2: Towards Chemical Foundation Models / Elana Simon et al Walid Ahmad // *arXiv preprint arXiv:2209.01712*. — 2022. — Pp. 1–8.

[2] *Sabrina Jaeger, Simone Fulle.* Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition / Simone Fulle Sabrina Jaeger, Samo Turk // *J. Chem. Inf. Model.* — 2020. — Pp. 27–35.

[3] Molecular contrastive learning of representations via graph neural networks / Yuyang Wang, Jianren Wang, Zhonglin Cao, Amir Barati Farimani // *Nature Machine Intelligence*. — 2022. — Pp. 1–9.

[4] *Jinhua Zhu, Yingce Xia et al.* Dual-view Molecule Pre-training / Yingce Xia et al Jinhua Zhu // *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. — 2023. — Pp. 3615–3627.

[5] *Rogers, David.* Extended-Connectivity Fingerprints / David Rogers, Mathew Hahn // J. Chem. Inf. Model. — 2010. — Pp. 742–754.

[6] *Weininger, David.* SMILES, a chemical language and information system / David Weininger // *J. Chem. Inf. Comput. Sci.* — 1988. — Pp. 31–36.

[7] SMILES-BERT: Large Scale Unsupervised Pre-Training for Molecular Property Prediction / Sheng Wang, Yuzhi Guo, Yuhong Wang et al. // *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. — 2019. `https://api.semanticscholar.org/CorpusID:202159174`.

[8] *Wu, Zhenqin.* MoleculeNet: A Benchmark for Molecular Machine Learning. — 2018.

[9] *Dwivedi, Vijay Prakash.* Benchmarking Graph Neural Networks. — 2022.

[10] PubChem Molecule Dataset. — `https://pubchem.ncbi.nlm.nih.gov/`. — Accessed: 2024-06-18.

[11] *Liu, Yinhan.* RoBERTa: A Robustly Optimized BERT Pretraining Approach. — 2019.

[12] *Klabunde, Ralf.* Daniel Jurafsky/James H. Martin, Speech and Language Processing / Ralf Klabunde // *Zeitschrift für Sprachwissenschaft*. — 2002. — 01. — Vol. 21.

[13] Recurrent Neural Networks (RNNs): Architectures, Training Tricks, and Introduction to Influential Research / Susmita Das, Amara Tariq, Thiago Santos et al. // Machine Learning for Brain Disorders / Ed. by Olivier Colliot. — New York, NY: Springer US, 2023. — Pp. 117–138. `https://doi.org/10.1007/978-1-0716-3195-9_4`.

[14] *Sennrich, Rico.* Neural Machine Translation of Rare Words with Subword Units / Rico Sennrich, Barry Haddow, Alexandra Birch // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). — Berlin, Germany: Association for Computational Linguistics, 2016. — . — Pp. 1715–1725. `https://aclanthology.org/P16-1162`.

[15] *Devlin, Jacob.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. — 2019.

[16] *AI, Facebook.* RoBERTa: A Robustly Optimized BERT Pretraining Approach. — https://huggingface.co/transformers/model_doc/roberta.html. — 2019. — Дата доступа: 11 июня 2024.

[17] *Kipf, Thomas N.* Semi-Supervised Classification with Graph Convolutional Networks. — 2017.

[18] *Veličković, Petar.* Graph Attention Networks. — 2018.

[19] *Maskey, Sohir.* Generalization Analysis of Message Passing Neural Networks on Large Random Graphs. — 2022.

[20] *Kim, Byung-Hoon.* Understanding Graph Isomorphism Network for rs-fMRI Functional Connectivity Analysis. — 2020.

[21] *Chengxuan Ying, Tianle Cai et al.* Do Transformers Really Perform Bad for Graph Representation? / Tianle Cai et al Chengxuan Ying // *Advances in neural information processing systems.* — 2021.

[22] *Landrum, Greg.* rdkit/rdkit: 2020_03_1 (Q1 2020) Release. — 2020. — . https://doi.org/10.5281/zenodo.3732262.

[23] The ChEMBL Database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods / Barbara Zdrazil, Eloy Felix, Fiona Hunter et al. // *Nucleic Acids Research.* — 2023. — 11. — Vol. 52, no. D1. — Pp. D1180–D1192. https://doi.org/10.1093/nar/gkad1004.

[24] *Loshchilov, Ilya.* Decoupled Weight Decay Regularization. — 2019.

[25] Automatic differentiation in PyTorch / Adam Paszke, Sam Gross, Soumith Chintala et al. — 2017.

[26] *Wolf, Thomas.* HuggingFace's Transformers: State-of-the-art Natural Language Processing. — 2020.

[27] *Biewald, Lukas.* Experiment Tracking with Weights and Biases. — 2020. — Software available from wandb.com. https://www.wandb.com/.

[28] *Buitinck, Lars.* API design for machine learning software: experiences from the scikit-learn project. — 2013.

[29] *Fey, Matthias.* Fast Graph Representation Learning with PyTorch Geometric. — 2019.

[30] *Seyone Chithrananda Gabriel Grand, Bharath Ramsundar.* ChemBERTa: Large-Scale Self-Supervised Pretraining for Molecular Property Prediction / Bharath Ramsundar Seyone Chithrananda, Gabriel Grand // *arXiv preprint arXiv:2010.09885.* — 2020. — Pp. 1–7.

[31] *Jean-Bastien Grill Florian Strub, et al.* Bootstrap Your Own Latent A New Approach to Self-Supervised Learning / et al Jean-Bastien Grill, Florian Strub // *Advances in neural information processing systems.* — 2020.

[32] *Bing Huang, O. Anatole von Lilienfeld.* Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity / O. Anatole von Lilienfeld Bing Huang // *J. Chem. Phys.* — 2016.

[33] *David L. Thakkar A., Mercado R.* Molecular representations in AI-driven drug discovery: a review and practical guide / Mercado R. David L., Thakkar A. // *J Cheminform.* — 2020.

[34] *Ye Hu, Dagmar Stumpfe.* Computational Exploration of Molecular Scaffolds in Medicinal Chemistry / Dagmar Stumpfe Ye Hu, Jürgen Bajorath // *J. Med. Chem.* — 2016. — P. 4062–4076.

[35] *Vijay Prakash Dwivedi, Xavier Bresson.* A generalization of transformer networks to graphs / Xavier Bresson Vijay Prakash Dwivedi // arXiv preprint arXiv:2012.09699. — 2020.