Отчёт

На основе нескольких статей, основная из которых — статья Qognitive о квантовой модели предсказания биржевого поведения, было решено разработать аналогичный подход. Несколько слов об этом исследовании:
Авторы в прошлом году разработали новый тип квантовой нейросети, особенно хорошо подходящий для задач определенного типа. Они в общих чертах описали его работу, а также сами задачи, которые решали. Одна из них — биржевая аналитика. Естественно, авторы сделали из этого также коммерческий продукт, поэтому информации было дано очень мало. Из-за этого значительную часть работы пришлось выполнить самостоятельно или с опорой на другие исследования. Также из-за этого потребовалось опробовать несколько методик, помимо основной, о которой будет упомянуто позже.

Суть задачи в том, чтобы научиться предсказывать динамику акций по биржевым и новостным параметрам (на основе статей FinBERT), **что не было реализовано у авторов оригинальной статьи**. Поэтому, прежде всего, следует описать, какие сырые данные были собраны для обучения.

Часть 1. Сбор и предварительная обработка данных

Были собраны биржевые данные с Мосбиржи. Мною был выделен пул из порядка 1500 топовых российских компаний. Для них были собраны дневные параметры на момент закрытия торгов за последние 5 лет. (На самом деле это даже многовато, так как считается, что связи, которые нас интересуют, меняются на среднесрочной перспективе, т.е. порядка 1-3 лет).
Включенные параметры:

- Цена закрытия акции
- Объём торгов
- Рыночная капитализация
- Выручка к рыночной капитализации
- Индекс Мосбиржи за день (глобальный параметр, позволяющий в дальнейшем устранить шумы)

Также были собраны новостные параметры.
Источниками послужили несколько новостных финансовых сайтов, точнее: Lenta.ru, РБК и «Коммерсантъ».
Был применен первичный фильтр по финансовым меткам.
Оказалось, что у РБК и «Коммерсанта» есть отличные выжимки самого важного из статьи, которые поддавались парсингу, например:

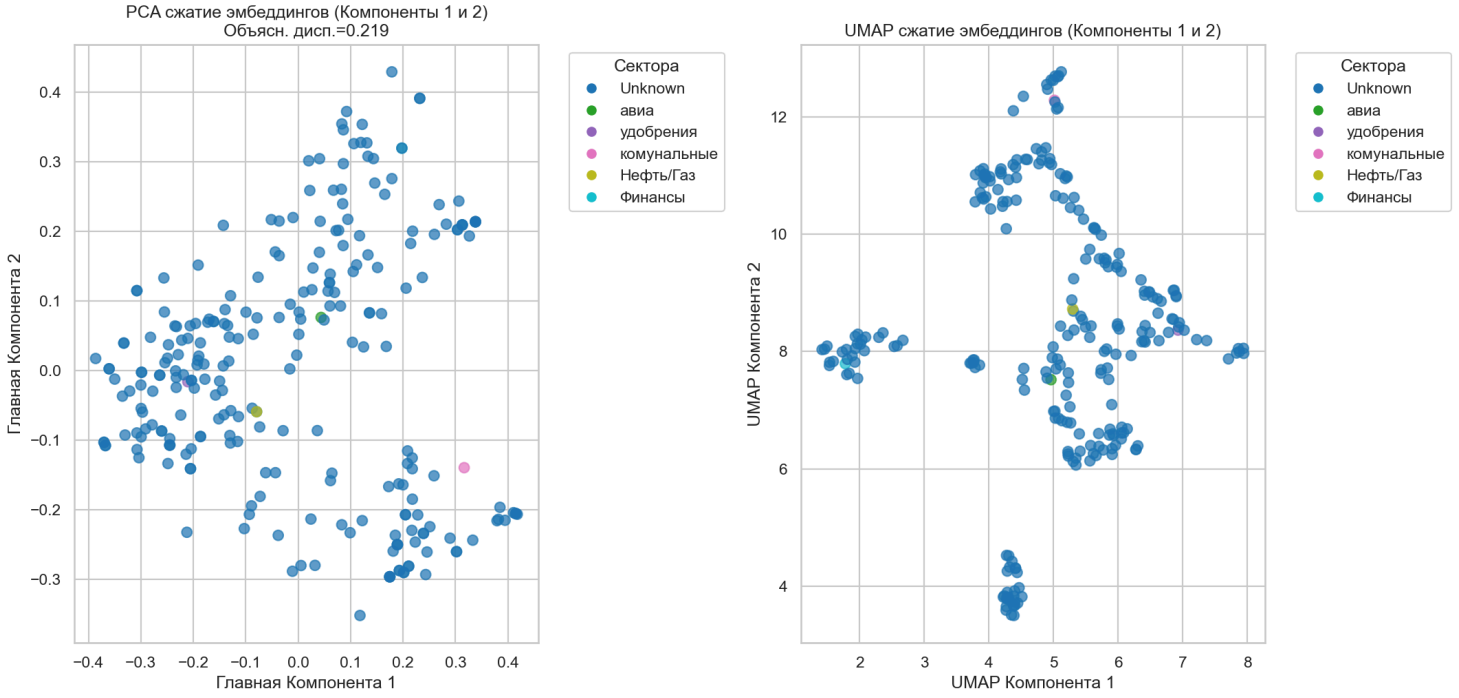
> "Австралийская TIG решила продать угольный бизнес в России", "TIG заключила юридически обязывающее соглашение о продаже добывающей компании «Берингпромуголь», управляющей компании «Берингуголинвест» и «Северной Тихоокеанской угольной компании», которая занимается геологоразведкой. Сумма сделки составила $49 млн."

Такие новости и их заголовки мной сразу отправлялись в классификационную модель Яндекса, которая возвращала нормированные показатели:

- Общий новостной сентимент и оценка положительности информации в категориях: макроэкономика России, монетарная политика, геополитика, нефтегазовый сектор, фискальная политика.

Количество таких новостей составляло около 10 в день, и они носили преимущественно глобальный характер.
Также использовался сайт Lenta.ru, где публиковались либо очень длинные тексты статей, либо короткие заголовки, освещавшие более узкие экономические темы; их количество достигало около 30 в день. Поэтому для их обработки использовалась LLM Gemini, откуда были получены те же самые параметры. Затем эти параметры были нормализованы и аккуратно объединены (простое сложение в данном случае неприменимо).

Видно, что новости — это глобальные признаки, не зависящие от конкретной акции. Поэтому, чтобы их связать, я спарсил описания компаний по их тикерам, а затем пропустил через эмбеддинг-модель с выходом в 3 измерения. Это решило проблему связи новостных признаков с отдельными акциями, а также просто дало акциям некоторую связь между собой.



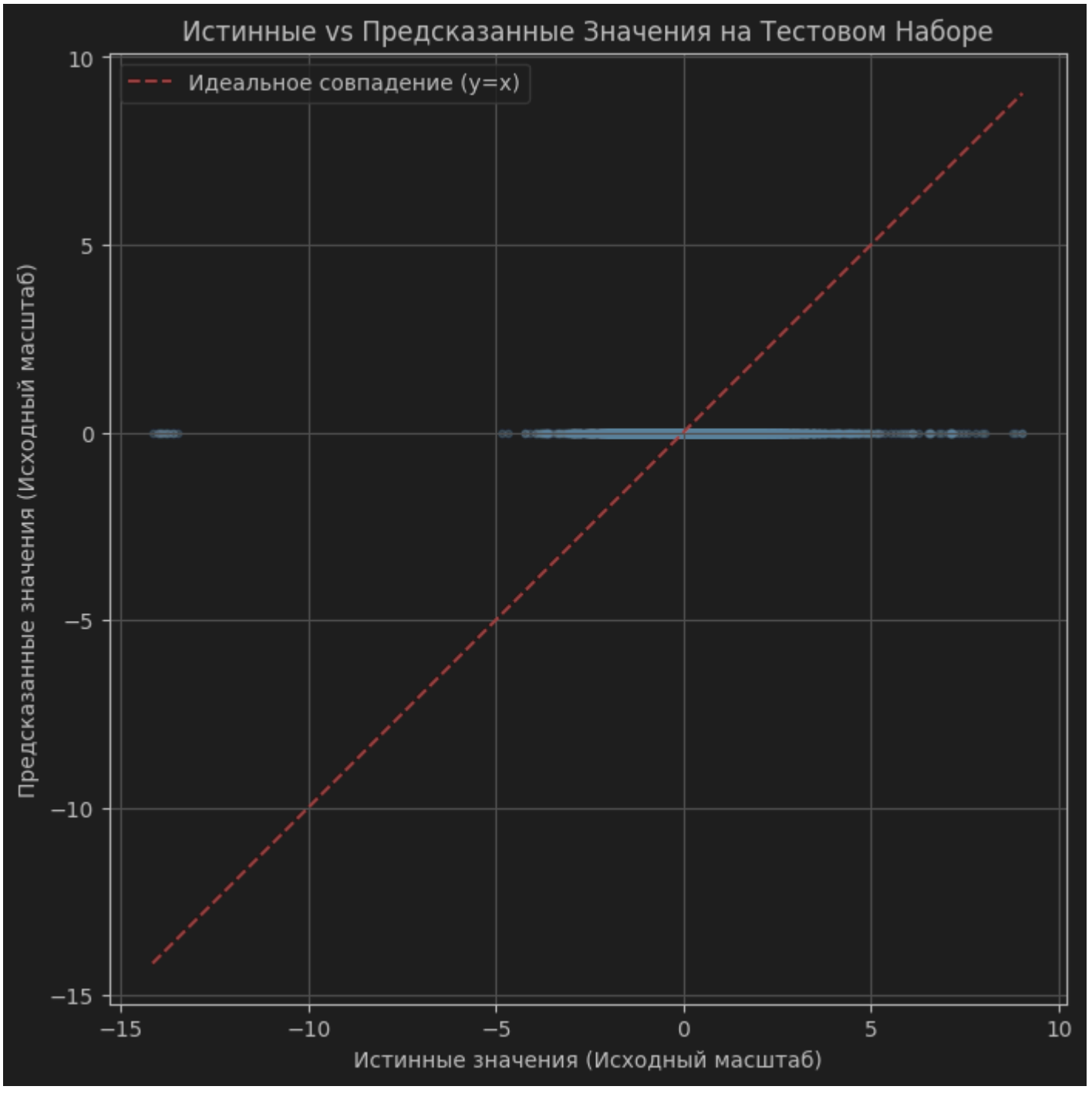
Часть 2. Первоначальные подходы и переход к авторской методике

В первый раз я неверно понял авторов, и, открыв статью, на которую они ссылаются, применил стандартный метод биржевых предсказаний, который практически не дал результатов.
Мой подход заключался в следующем: Я решил не использовать временные данные, чтобы не усложнять модель и не искать слишком очевидные корреляции. Поэтому были взяты данные не для отдельных компаний, а параметры 5 стандартных биржевых индексов + упомянутые новости. Затем из этих параметров были вычислены различные временные деривативы (желательно не дублирующие друг друга), вроде средневзвешенной волатильности каждого индекса за неделю, различные Z-усреднения за разные промежутки времени. Ожидалось, что это позволит модели сориентироваться в общебиржевой обстановке, не засоряя параметры

временными рядами. Но результата это почти не дало: классические нейросети были беспомощны, вероятно, из-за очень малого количества параметров. На порядок более сложные классические и квантовые модели дали минимальный результат, который соответствовал, как будет ясно позже, окололинейным связям.

Типичные результаты первой модели:

```
--- Результаты оценки (CNN-GRU) ---
MSE NN: 118.9892
RMSE NN: 10.9082
MAE NN: 8.4029
R² NN (vs Mean): 0.0340
---
```



Теперь рассмотрим авторский подход:
Представим себе, что можно единовременно смотреть все дневные показатели на бирже. Ясно,

что все эти данные можно попробовать связать между собой простейшим линейным способом: можно сделать стандартную линейную регрессию по всему набору этих параметров, предсказывая **доходность** каждой акции.

В данном случае к этому добавляются также новостные параметры.
Имеется вектор Y, состоящий из доходностей всех акций за день, и матрица X со всеми данными, кроме эмбеддингов (по ним нет смысла проводить регрессию), которые были предварительно кросс-секционно нормализованы и пересчитаны в более удобные параметры. Затем проводится OLS-регрессия (по сути, МНК), из которой получаются остатки доходности, не объясняемые линейной компонентой. Теперь это будет наш целевой параметр.

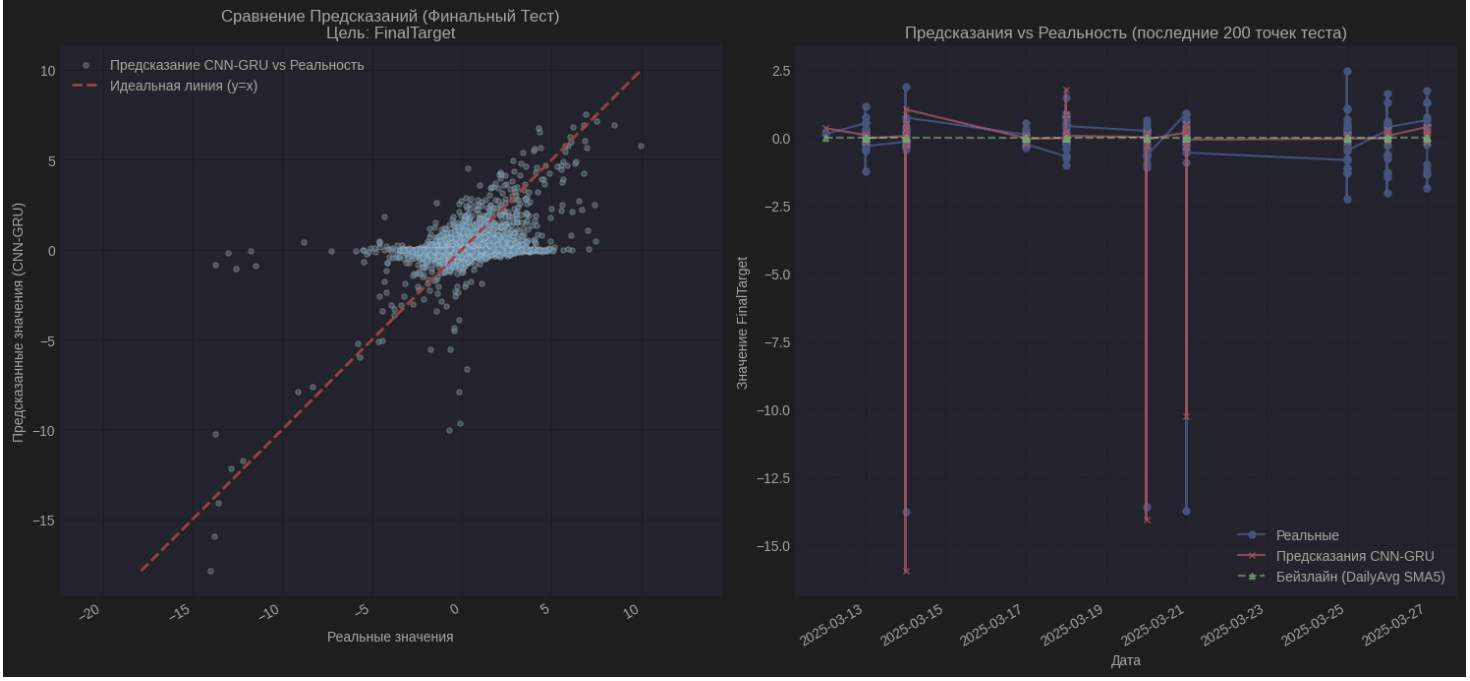
Это было сделано по следующим причинам:

- Я планировал воспользоваться преимуществами квантовой нейросети и найти именно **нелинейные** сложные зависимости.
- Теперь я могу подавать на вход нейросетям только глобальные параметры + параметры отдельной акции, так как их линейные связи уже очищены и контекст других акций не так важен. Это **радикально снижает** количество параметров, соответственно, появляется возможность обучить более глубокую модель.
- На рынке уже есть фонды с огромными моделями, причем модели этих игроков содержат в миллионы раз больше параметров, с данными, которые мне даже недоступны. Но чаще всего они считают что-то около линейной регрессии. Поэтому, отсекая линейные части, появляется возможность получить преимущество перед существующими фондами, что и создает возможность для извлечения прибыли на рынке. Кроме того, такие фонды занимаются выравниванием курса на эту линейную компоненту (проще говоря, арбитражем), косвенную информацию о котором мы получаем и действительно исключаем.

Теперь о самих моделях.
Классическая модель, с которой сравнивались авторы, практически не дала результатов. Сколько-нибудь значимые результаты были получены только начиная с архитектуры:

```
CNN_OUT_CHANNELS = 32
CNN_KERNEL_SIZE = 3
GRU_HIDDEN_SIZE = 64
GRU_NUM_LAYERS = 1
MLP_HIDDEN_SIZE = 32
```

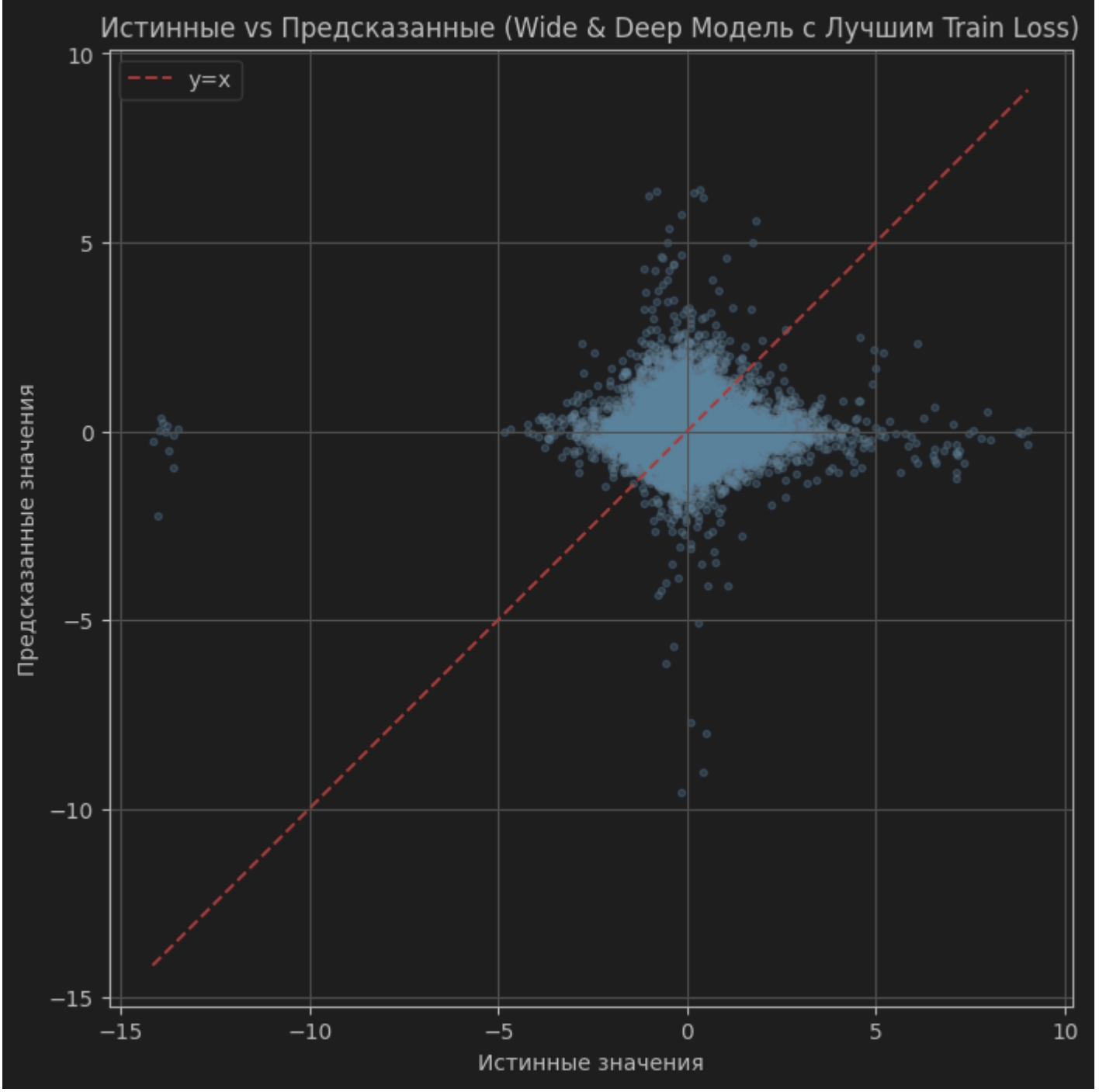
что привело к $R^2 \approx 0.2$.

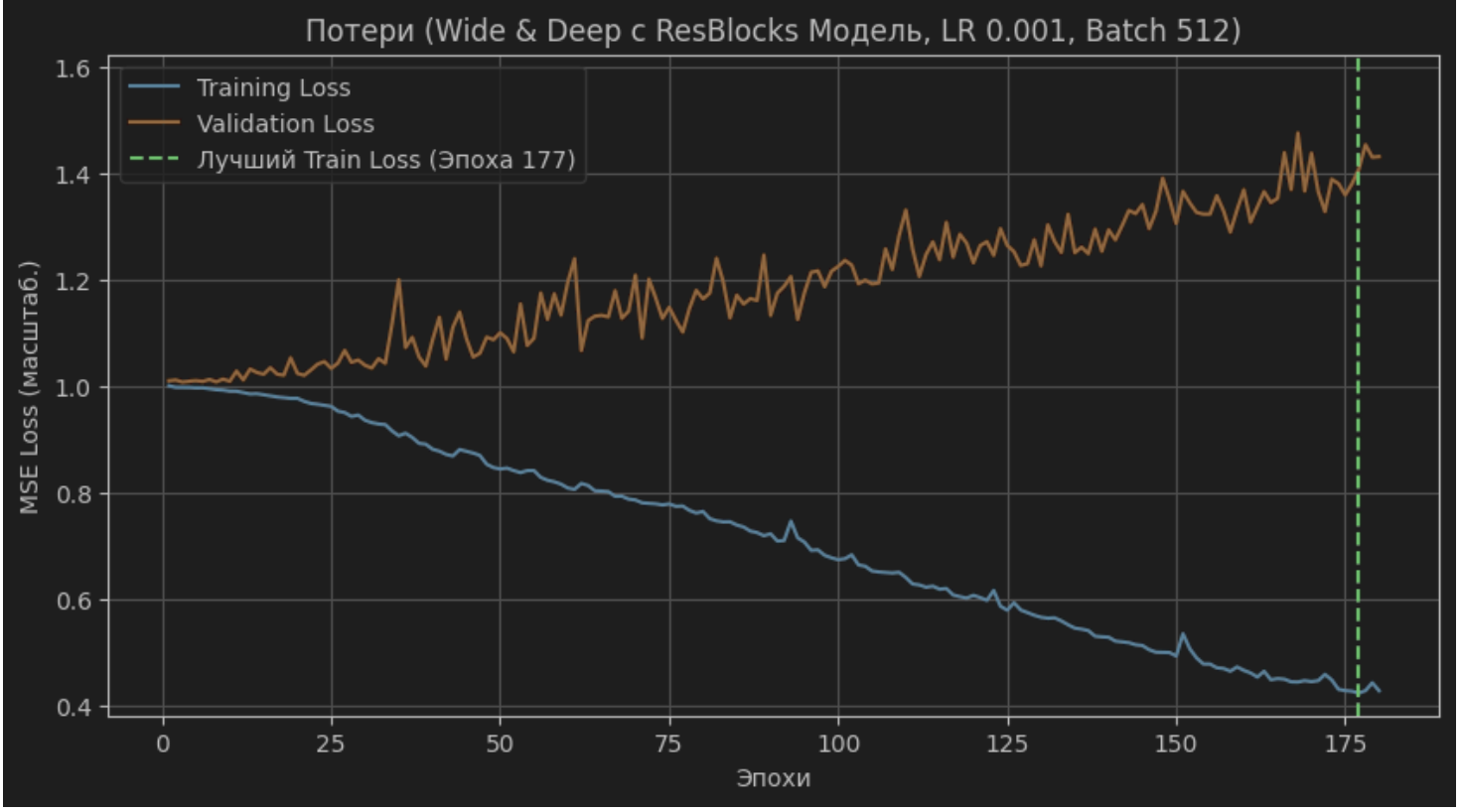


Мои попытки воспроизвести этот результат не увенчались полным успехом (был получен $R^2 \approx 0.086$), поэтому, возможно, в оригинальном исследовании имела место утечка данных. Однако воспроизводимый результат для этой модели также демонстрирует определенную эффективность.

В любом случае, это:

- доказывает, что связи в этой очищенной модели действительно есть, и можно получать стабильные положительные результаты (которых, конечно, все еще недостаточно, чтобы "переиграть биржу");
- этот более сложный подход начал учитывать связи между днями (GRU). Хотя этого хотелось избежать, так как ищется другая зависимость, это все еще не временные ряды в прямом смысле, так как дата как параметр нигде не подавалась.


Истинные vs Предсказанные (Wide & Deep Модель с Лучшим Train Loss)
y=x
Предсказанные значения
Истинные значения
10
5
0
−5
−10
−15
−15
−10
−5
0
5
10



Часть 3. Квантовая модель по концепции авторов

Авторы представили концепцию квантовой модели, работающей следующим образом:

Имеется вектор с данными X и целевой параметр Y.
Также имеется некоторый гамильтониан ошибки, зависящий от X и Y.
Также задается состояние φ(n, P1), где n — число кубитов в системе, а P1 — набор некоторых параметров.

Шаг 1.
Фиксируется H(X, Y=none) и считается Error(P1) = <φ| H(X) |φ>. Теперь просто проводится оптимизация по P1, уменьшая ошибку — так находится так называемое "базовое состояние φ". То есть, исходя из наших данных, создается состояние, которое должно максимально точно описывать КОНКРЕТНЫЕ параметры.

Шаг 2.
Ясно, что пока этот гамильтониан ничего на самом деле не значит, и необходимо минимум его ошибки при конкретных данных свести к тому, что было что-то предсказано. Теперь задаются предсказания — это некоторые операторы для каждого параметра и для целевого значения. Пусть это некоторые A_x_i и A_y, т.е. предсказание x_i = <φ| A_x_i |φ>.
Тогда классический гамильтониан ошибки будет иметь вид:

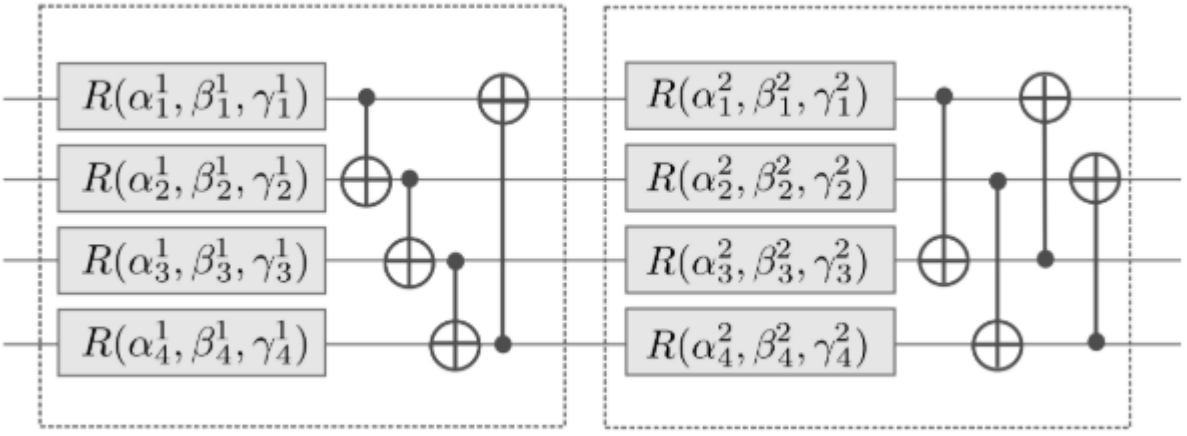
$$\sum_k \left(\hat{A}_k - \mathbf{x}_{t,k} \cdot I\right)^2$$

Соответственно, чтобы его натренировать, необходимо изменять A_{x_i} и A_y, которые некоторым образом параметризованы. Это делается градиентным спуском. Получается, что второй шаг — это тренировка по второй группе параметров P2, которая снова уменьшает <φ| H(P) |φ>.
После спуска изменяются P2 (которые и являются весами модели), P1 же каждый раз находятся заново.
Затем алгоритм повторяется по всему датасету.

Предсказание:
Чтобы сделать предсказание, нужно в точности повторить первый шаг: найти базовое состояние, а затем просто посчитать <φ| A_y |φ>, который, как было определено ранее, является предсказанием величины y при поданном X.

Реализация.
Основной вопрос — как параметризовать эти самые операторы (матрицы размера 2^n), при условии, что операторы должны быть эрмитовыми.
Конечно, это можно сделать через какую-то композицию эрмитовых операторов с параметризованными коэффициентами, но это не то, что естественно переводится в реальную квантовую схему. Поэтому я решил, что нужно строить оператор как некоторую квантовую схему. Так как она состоит из унитарных гейтов, то проще всего сделать из нее что-то эрмитово: U† B U = Âk. Где U — параметризованная квантовая схема, а B — простой эрмитов оператор (в моем случае это была просто диагональная матрица с различными числами для получения различных собственных чисел).
Таким образом, модель была действительно правдоподобно параметризована через квантовую цепочку, но платой за это стало то, что больше 2/3 всех гейтов не параметризуются, а выполняют "подгоночную функцию".

Все это я решил реализовать на знакомом мне и, как показалось, подходящем фреймворке PennyLane.
И квантовое состояние φ, и параметризованные U я сделал как цепочки pennylane.templates.layers.StronglyEntanglingLayers.



Просто с разным количеством слоев: от количества слоев φ зависит дискретизация полученного "слепка" входных параметров, от количества слоев U зависит глубина модели.

Гамильтонианы я реализовал через встроенный класс гамильтониана, спуски реализовал стандартным методом adjoint.

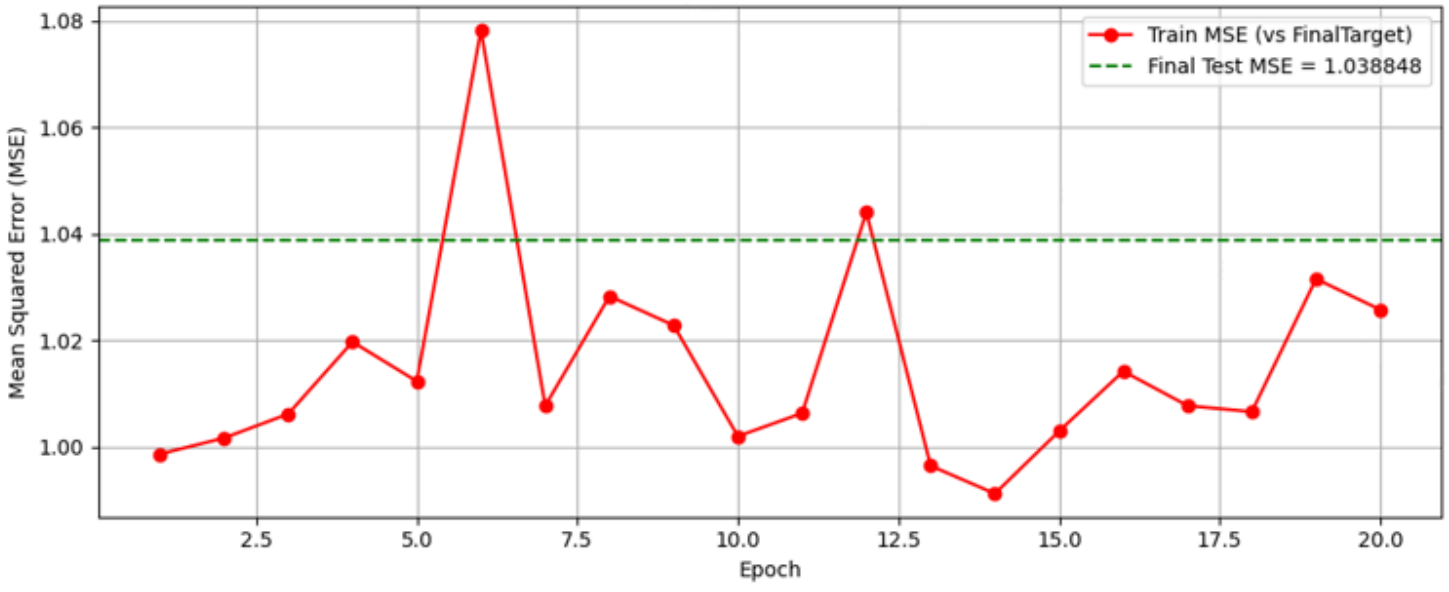
Оценка количества параметров:
При 19 фичах, 5 кубитах, layers=3 получается 19x5x3x3 — порядка 1000 параметров.

При первом же обучении я столкнулся с проблемой — схемы при пяти кубитах хорошо считаются и на CPU, переход на GPU не дает никакого прироста, даже увеличивает время из-за накладных расходов. Но эта пятикубитная схема запускается огромное число раз, из-за этого время обучения становится огромным. Я научился параллелить процесс на CPU на число ядер, и на 24 ядрах смог получить результаты в районе 6 часов на эпоху при минимальном количестве слоев. Ясно, что проблема не в какой-то огромной сложности этой модели, а в ее (недостаточной) оптимизации. Я долгое время пытался оптимизировать эту схему, переписав большую часть кода на PyTorch, а также перейдя на ускоренное разложение гамильтониана на матрицы Паули, что позволило добиться трехкратного ускорения до 2 часов на эпоху.

Варианты решения, которые я вижу:

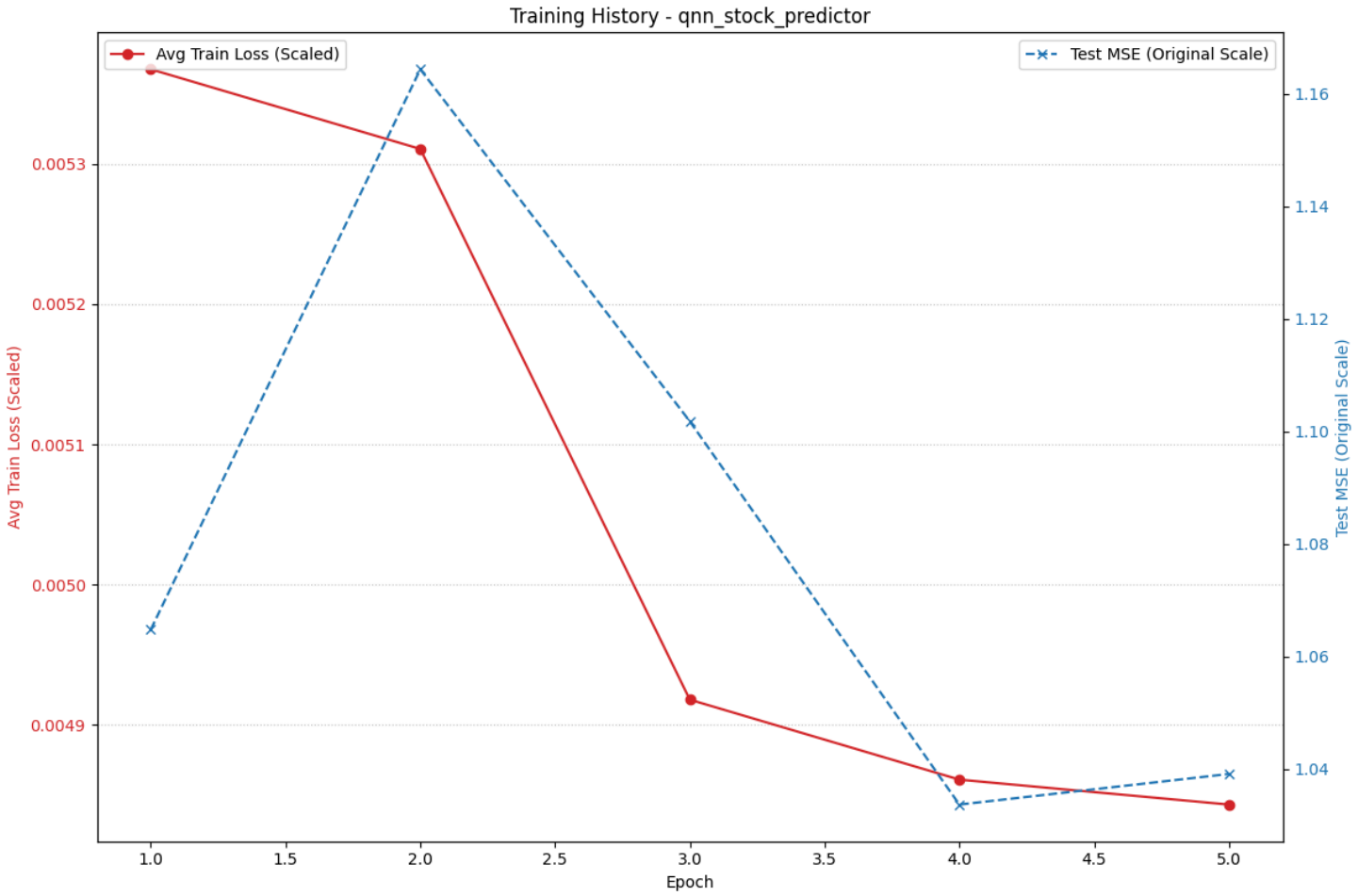
- Запуск модели на каком-то огромном числе CPU, что, насколько я знаю, вполне реально и стоит сравнимо с арендой GPU-кластера, но найти такую конфигурацию мне не удалось.
- При увеличении числа кубитов и запуске на GPU скорость обучения не особо меняется, так как вступает в силу ускорение GPU. В таком случае затрачиваемые мощности будут вполне приемлемы.
- Переписывание модели на другой фреймворк с оптимизацией под GPU.

Также я пробовал уменьшать в разы число данных (в 10 раз).



Но, как видно, результатов это совсем не дало.

Несмотря на это, я несколько раз потренировал эту модель (порядка 5 эпох) и убедился в том, что модель на этих данных уже после первой эпохи достигает train loss порядка 0.1, который затем продолжает опускаться, вместе с метрикой MSE.



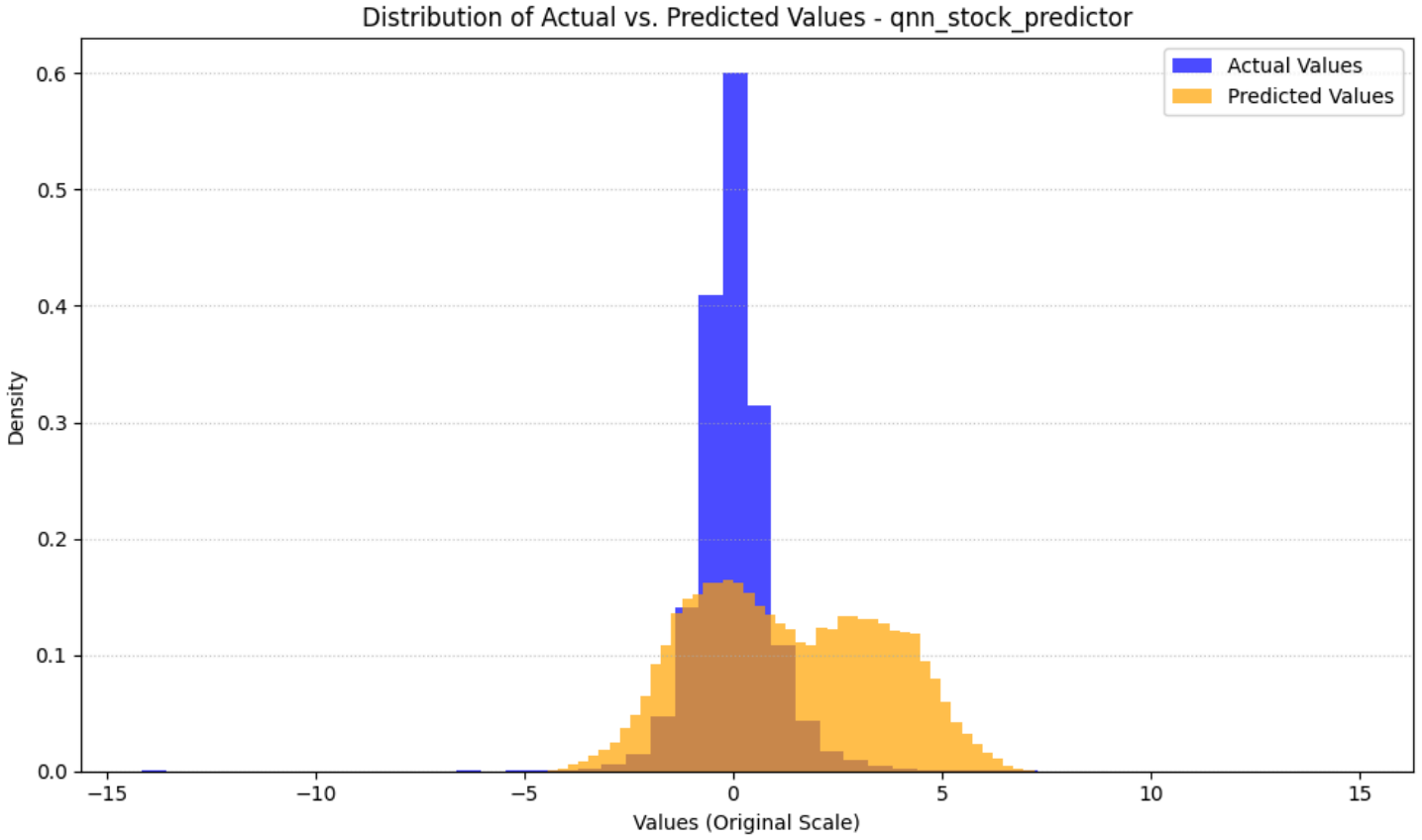
К сожалению, больше ничего из этой модели получить я не успел из-за временных ограничений.

Часть 4. Альтернативная квантовая модель

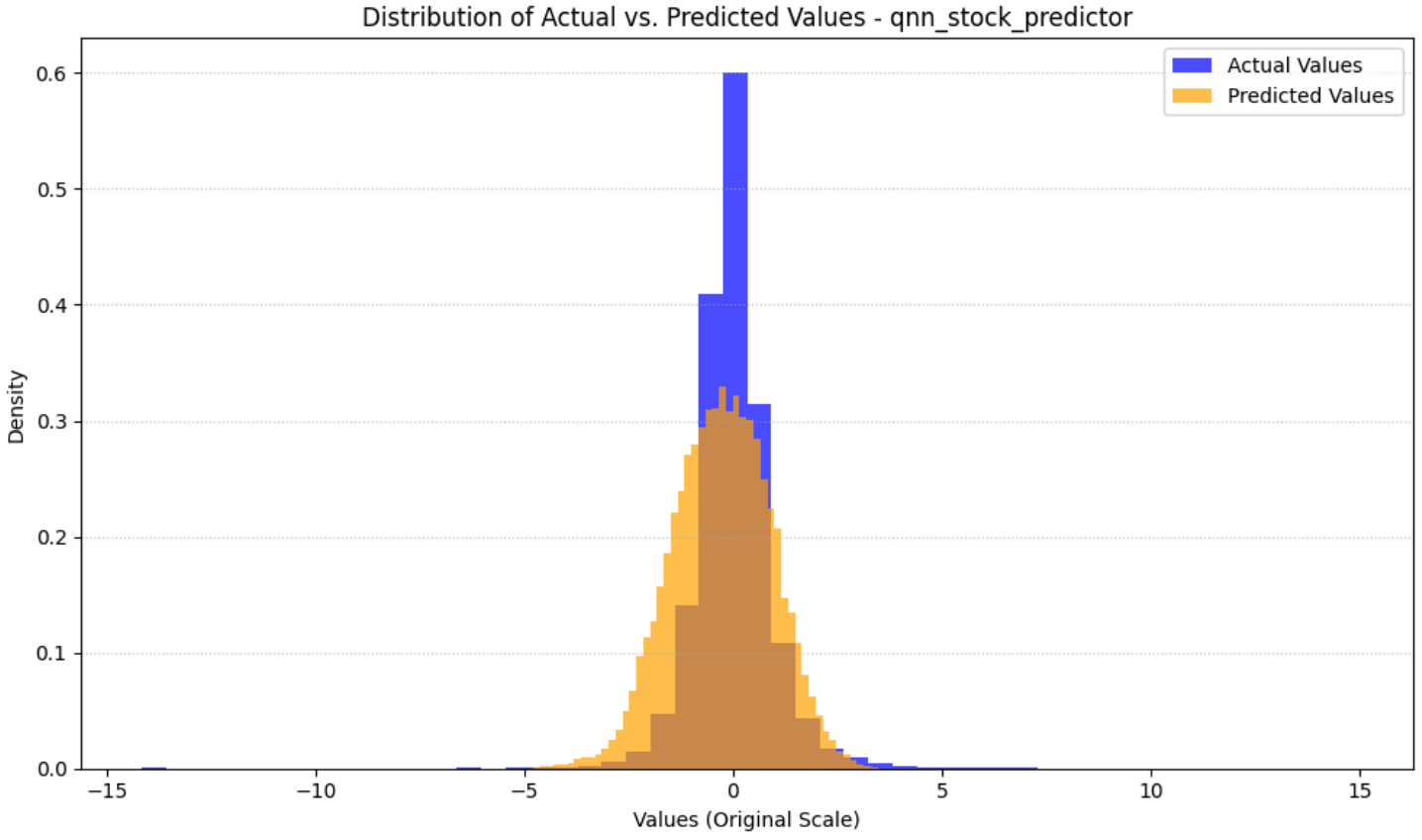
Тогда я решил реализовать третью, более простую квантовую модель для задачи нелинейного поиска.
Модель также состояла из слоев pennylane.templates.layers.StronglyEntanglingLayers. Их число я варьировал от 1-2, чтобы не допустить переобучения уже на первых порах.
Число кубитов я изначально выставлял в соответствии с числом параметров, но такая модель тренировалась слишком долго даже на GPU (долго, чтобы успеть с ней поработать, хотя это абсолютно ожидаемое время для такой модели).
Тогда я сделал сжатие состояния до 5 кубитов (2^5 > 18), что, конечно, уменьшало разрешающую способность модели, но это вполне соответствовало задаче поиска нелинейности, интерпретируя это как первый сжимающий слой. Также я это компенсировал увеличением числа слоев до 3-6.

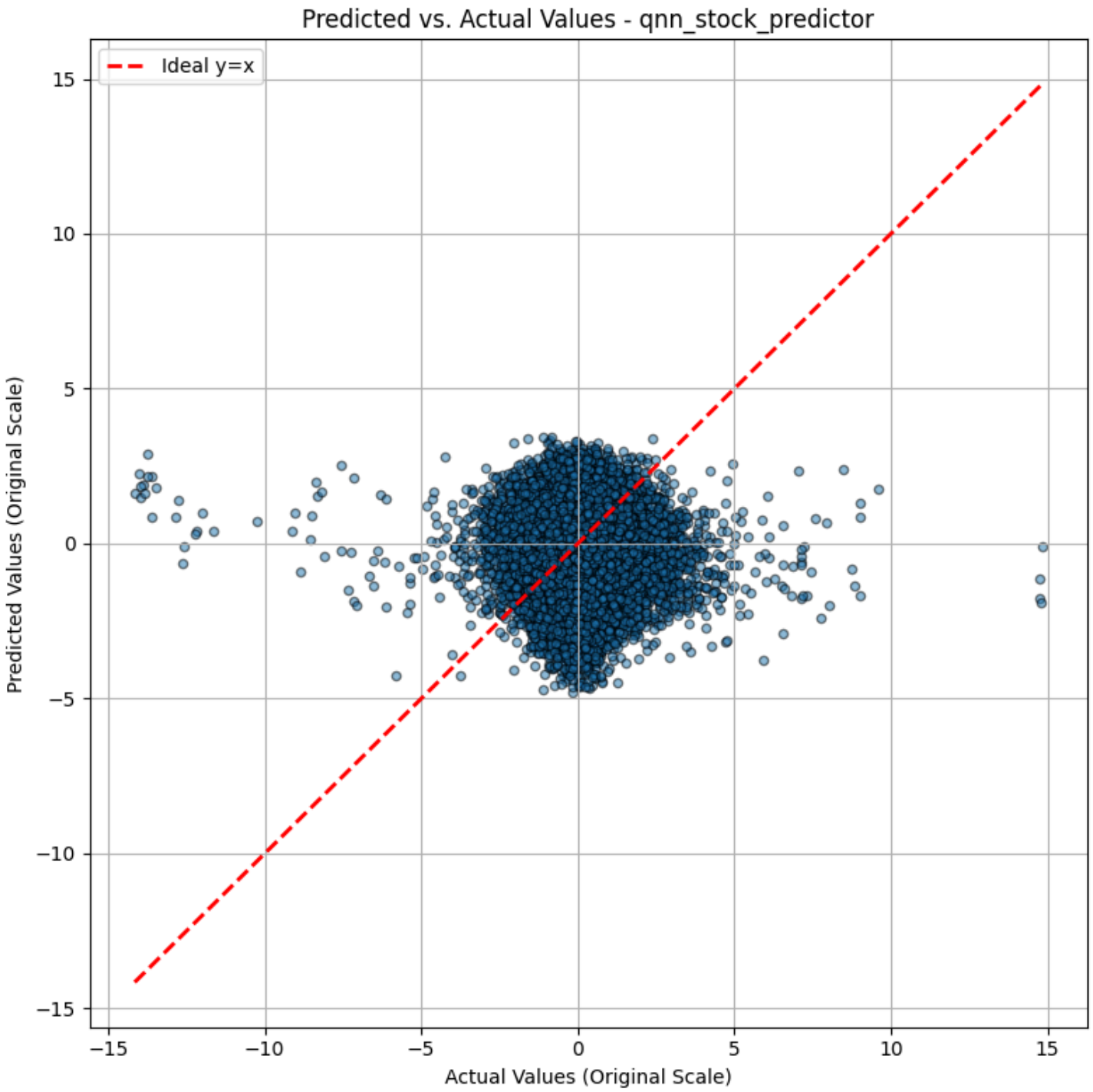
Здесь я уже получил первые адекватные результаты.

Для 6 слоёв и 3 эпох:

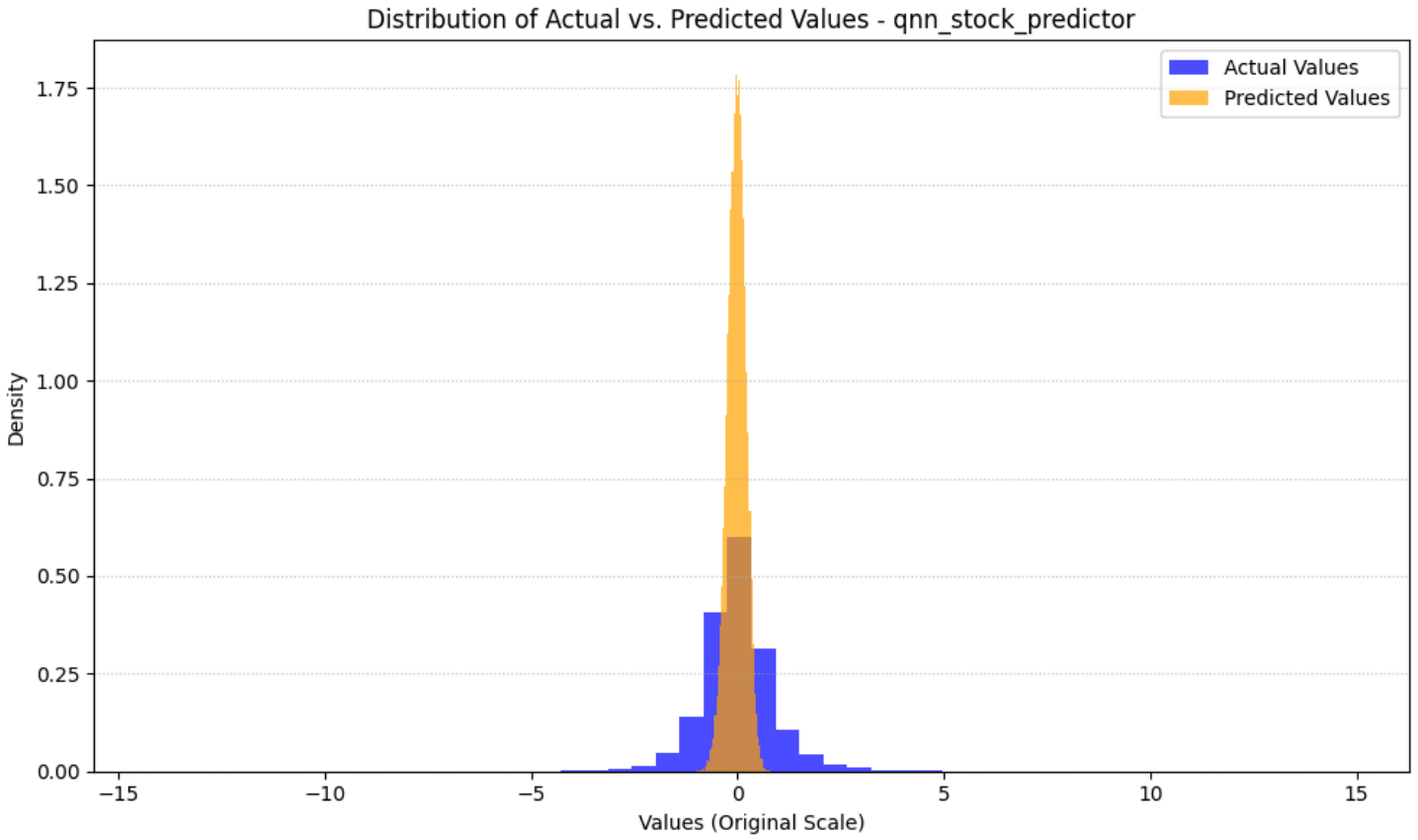


Для 6 слоёв и 8 эпох:





Для 4 слоёв и 6 эпох:



Здесь видно, что модель действительно учится предсказывать экстремальные значения даже при малом числе слоев (хоть и хуже).

Интерпретация данных:
Проблема с интерпретацией результатов: модель сложно оценить из-за биржевой специфики и специфики целевого параметра. Даже если бы предсказывалась чистая доходность, то результат MSE=0.8 мог бы считаться отличным, так как в большинстве случаев он находится внутри рыночных колебаний, но при этом видит паттерн, следуя которому получается прибыльная стратегия. То же самое справедливо и в моем случае. Более того, предсказывается мной не просто доходность, а именно нелинейная компонента — компонента "недооцененности актива" стандартными алгоритмами. Если бы я оценивал эффективность полученной модели, то делал бы это следующими способами:

- как авторы, попытался бы вычесть из результатов классической модели квантовые предсказания и показал бы явные нелинейные связи;
- построил бы обобщенную финансовую стратегию на основе получаемых индексов, а затем прогнал бы её по историческим данным;
- ввёл бы более сложную метрику потенциальной прибыли для предсказанных значений;
- усреднял бы предсказания по некоторым группам компаний и связывал бы их доход с недооцененностью похожего по структуре фонда — это простейший способ применения такой модели.

Для меня результатом стала эффективность снижения train loss в квантовой модели по сравнению с классикой, а также наблюдение сложных нелинейных и вневременных компонент, которые не обнаруживались на классике.

Что не удалось реализовать (из-за нехватки времени):

- Собрать отчетные данные компаний с их доходностями (как у авторов). Это могло бы сильно улучшить предсказания, но для этого потребовалось бы искать эти отчетности для каждой отдельной компании.
- Провести полноценное обучение моделей.

Материалы:

- Код для сбора датасета находится в папке news_parser.
  - Подпапка dataset2version содержит парсинг биржевых данных, обработку, слияние с новостным датасетом, эмбеддинг, финальный регрессионный фильтр.
  - Новостной анализ находится в папке analysis.
- В папке QCML есть разделы для всех моделей. Сама архитектура моделей описана в файлах qcml_model.py и quantum_model.py.