Отчет о менторском проекте

Название проекта: Integrating Quantum SVD & LSTM for Improved Recommender Accuracy
Автор: Артем Чурилкин
Менторы: Асель Сагингалиева, Арсений Сенокосов

1 Введение и цели

В рекомендательных системах матричные подходы (SVD, NMF, ALS, etc.) используюся уже давно и дают стабильные результаты в практических реализациях, но подобные методы ограничены по точности и вычислительным ресурсам (хоть и допускают распараллеливание процессов), особенно на больших объемах данных современных сервисов, где пользователей и контента становится все больше и больше с каждым днем. Квантовые алгоритмы (в частности, квантовый SVD) теоретически позволяют работать быстрее (время полилогарифмическое от размера данных). Правда такие подходы требуют тщательной проверки, поскольку текущие реализации на симуляторах пока еще не идеальны и не всегда оправдывают теоретические ожидания. Здесь естественным образом возникает вопрос, а есть ли в реальности преимущество у квантового подхода над проверенными классическими методами, и можно ли это преимущество усилить, используя гибридные схемы.

- Конкретная задача: предсказание рейтингов в датасете MovieLens100k, где предложены явные пользовательские оценки фильмов по шкале от 1 до 5.
- Что хотелось получить: честное сравнение гибридного метода Quantum+LSTM с классикой.
- Главные задачи проекта:
  - Провести честное сравнение эффективности классических методов (Truncated SVD, NMF, ALS) с квантовым методом (вариации Quantum SVD).
  - Разработать и проверить гибридную схему Quantum SVD + LSTM, чтобы корректировать ошибки квантовых предсказаний и, возможно, найти скрытые паттерны, которые не видят классические подходы.
  - Понять оправдано ли использование QML для RecSys сейчас, и есть ли в подобных задачах практическая польза от комбинирования квантовых методов с нейросетями.

2 Краткая сводка проекта

В проекте реализован классический подход к рекомендациям (Truncated SVD, ALS) и квантовый подход на базе Pennylane, а затем и гибридный подход с коррекцией результатов квантовой части с помощью LSTM.

- **Данные и baseline:** MovieLens100k разделён на train/valid/test (60/20/20%), рейтинги нормированы в $[0, 1]$ и представлены разреженными CSR-матрицами. В качестве классических подходов имеем Truncated SVD (sklearn) и ALS (самописный по Яндекс-хендбуку). Подбор гиперпараметров реализован через Optuna.
- **Квантовый SVD:** Раз нет возможности использовать QRAM, то данные встраиваются классическим TruncatedSVD$\to U, V^\top$ и затем кодируются двумя независимыми схемами (по итогу для гибридного подхода выбрана схема №2):
  1. SWAP-тест (AmplitudeEmbedding + CSWAP) для оценки скалярного произведения;
  2. Вариационная сеть с AngleEmbedding, L слоёв RY+CNOT, измерение $\langle Z_0 \rangle$ и линейная корректировка (s, b).

  На 20 000 предсказаний требуется ∼270 с, что является очень затратным по времени в сравнении с классическими подходами. При этом второй подход выдает значения в весьма узком диапазоне [3.40, 3.64], то есть около среднего значения оценки в датасете.

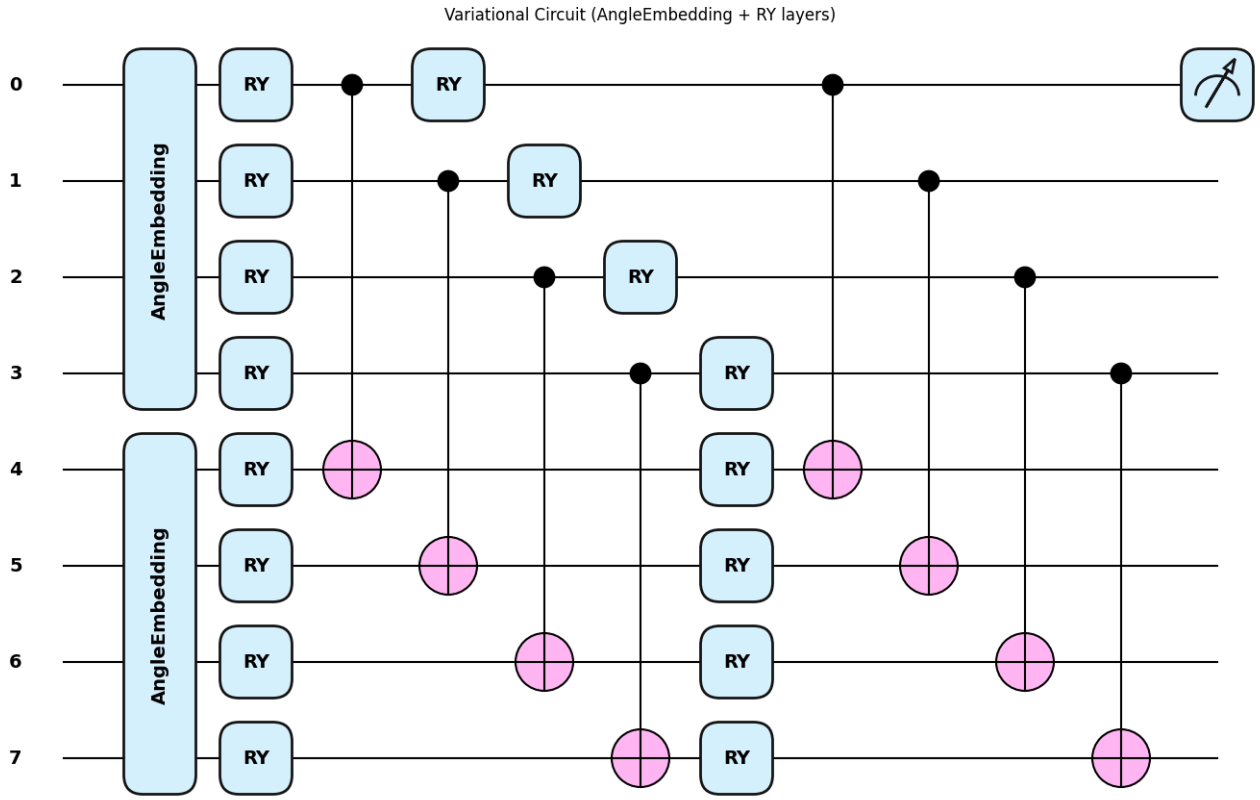

Рис. 1: Схема №2, используемая для гибридного подхода

- **Гибрид Quantum+LSTM:** На вход LSTM-регрессору подаются предсказания квантовой (AngleEmbedding) сети r_q, ID пользователя и ID фильма, затем он обучается прогнозировать ошибку $e = r_{\text{true}} - r_{\text{q}}$. Итоговый рейтинг $\hat{r} = r_{\text{q}} + \hat{e}$ позволяет расширить диапазон рекомендаций (предсказанная ошибка $\in [-3.2, 2.1]$), добавляя к инференсу лишь несколько сотен миллисекунд. При этом RMSE поменялся незначительно (с 1.12 в квантовом случае на 1.09 в гибридном).

3 Выводы

- ALS показал наилучшее сочетание качества и скорости: RMSE ≈ 0.70, MAE ≈ 0.31 при инференсе за 0.18 с. Так что по чистым цифрам обойти классические методы с помощью выбранных квантовых подходов не удалось.
- Квантовые схемы (AmplitudeEmbedding, AngleEmbedding + вариационная сеть) работоспособны, но требуют сотни секунд на 20 000 предсказаний. Они генерируют оценки в узком диапазоне около среднего значения, что означает не совсем успешную работу алгоритма. По сути выходит, будто алгоритм стремится в лоб уменьшить метрику, не учитывая индивидуальные предпочтения пользователя и выдавая среднюю оценку для всех.
- Гибрид AnglEmb+LSTM расширяет диапазон квантовых предсказаний, при этом незначительно снижает RMSE (с ~ 1.12 до ~ 1.09). Фактически это является улучшением с точки зрения индвидиуальных предпочтений пользователя (ведь диапазон предсказаний расширяется, для каждого пользователя оценки становятся более разнообразными, а, значит, более близкими к действительности), а не просто угадыванием средней оценки, как было в квантовых схемах. Это подтверждает мысль о том, что не всегда голые цифры являются показателем успеха и работоспособности моделей.
- Подход с обучением на ошибке квантовых предсказаний оказался недостаточно информативным – модель предсказывает линейный сдвиг, а не сам рейтинг, что ограничивает потенциал поиска нетривиальных (скрытых) зависимостей.

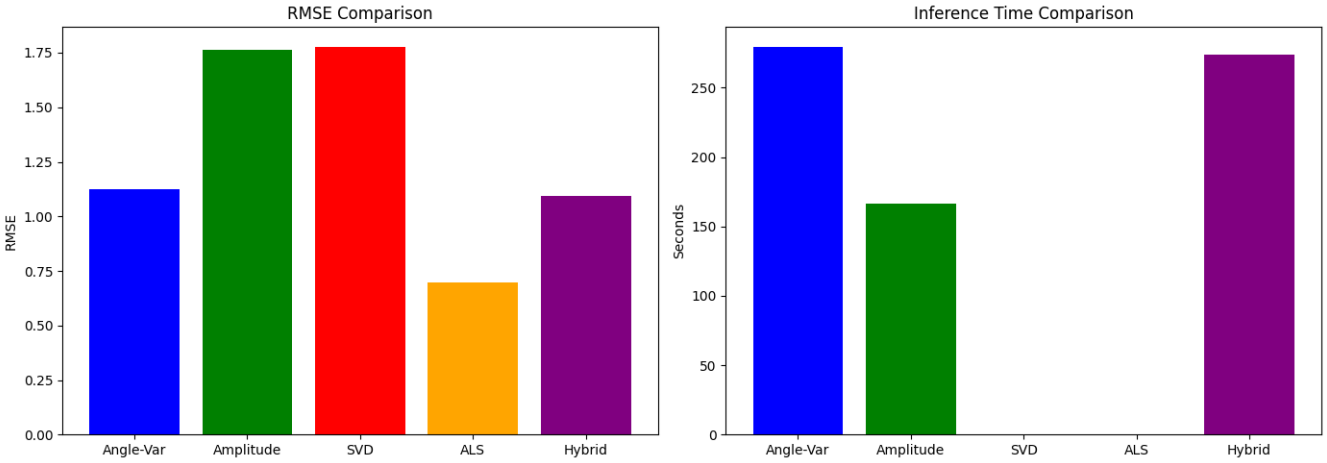

Рис. 2: RMSE на тесте и время инференса. ALS оказался лучшим.

4 Дальнейшие возможные шаги

- **Батчинг и параллелизация QNode:** реализовать пакетную обработку запросов в Pennylane и распределить симуляцию по нескольким потокам или узлам, чтобы ускорить время работы и стать ближе к уже готовым распараллеленным классическим методам.
- **Оптимизация квантовой схемы:** изменить топологию (например, сократить глубину и количество CNOT-слоёв), присмотреться к квантовым автоэнкодерам.
- **Пересмотр задачи LSTM-коррекции:** обучать сеть напрямую на предсказании окончательного рейтинга $\hat{r}$ вместо ошибки, а также протестировать альтернативные архитектуры (GRU, Transformer) и loss-функции (MAE, Huber). Вполне возможно, что при таком подходе получится уловить больше скрытых зависимостей и улучшить результат.
- **Масштабирование и проверка на реальном QPU:** оценить подход на больших датасетах (MovieLens1M, Netflix) и проверить квантовую схему на облачном QPU для измерения реального времени и уровня шума.
- **Холодный старт и ансамблирование:** использовать квантовые латентные представления для холодного старта новых пользователей/фильмов и создать ансамбли ALS + Quantum + ML-коррекция для повышения устойчивости и точности.

5 Статьи, вдохновившие на написание проекта:

- Ключевая статья (Kerenidis–Prakash) — в ней предложено значительное ускорение по обработке матриц (это прям такая основа, от которой отталкивался)
- Улучшение точности с LSTM архитектурой (Yikai Mao, Shaswot Shresthamali, Masaaki Kondo) — свежая статья, благодаря которой вдохновился вопользоваться LSTM поверх квантового SVD, чтобы ошибки предсказаний корректировать.
- Оверьвю (Giovanni Semeraro & Marco Polignano) — описание основных методов на 2022 год по рекомендациям, было полезно для общего развития и понимания происходящего в квантовых методах RecSys.
- P.S. Уже после того, как начал писать этот отчет, наткнулся на интересную статью от Skoltech и Sber AI Lab, в которой рассказывается про сравнение различных методов в RecSys. На Figure8 можно увидеть, что EASE круче ALS, так что можно было бы сравнивать квантовые алгоритмы и с этим методом.

6 Appednix (технические детали)

Основной отчет законен на 5 пункте, далее детали для тех, кто хочет сильнее погрузиться или узнать про исользуемые матричные методы подробнее, а также ближе познакомиться с архитектурой используемых моделей.
Ноутбук с кодом можно найти по ссылке.

6.1 Обработка датасета и baseline

Для работы с классическими методами сделал предобработку данных: перешел на работу с разреженными матрицами, потому что сам по себе MovieLens100k весьма разрежен. Изначально использовал нормализацию строк по L2, но возникла проблема — модели воспринимали нулевые значения как низкие оценки, что приводило к искажению результатов (сильно снижались значения метрики RMSE, что на деле не являлось честным значением, а было только проблемой предобработки данных). Поэтому от L2 отказался и решил просто сделать перемасштабирование из $[1, 5]$ в $[0, 1]$. Датасет разделил на три части: train (60%), valid (20%) и test (20%). Матрицы взаимодействий представил в виде разреженных матриц формата CSR (Compressed Sparse Row), что оптимально по памяти и вычислительно эффективно. Запустил эксперименты с Truncated SVD (sklearn) для подбора гиперпараметров через Optuna. Параметры подбирал по лучшему значению RMSE на валидационной части, но заодно фиксировал MAE и время выполнения каждого, чтобы потом сравнивать инференс моделей между собой.

Использование NMF (Non-negative Matrix Factorization) из sklearn не задалось, поскольку снова вылезла проблема с обработкой нулевых элементов, но уже в другом ключе: встроенный в библоитеку воспринимал 0 как информативное значение, а маску на игнорирование этих нулей даже в разреженной матрице наложить не удалось. Из-за этого результаты были некорректны, и метод не давал возможности эффективно подбирать гиперпараметры (значение метрики практически не менялось вне зависимости от параметров). По итогу от NMF отказался и перешел к ALS (Alternating Least Squares). Основная идея ALS — попеременная оптимизация латентных факторов пользователей и объектов путём решения задач наименьших квадратов с регуляризацией (L2-регрессия). Реализацию написал с нуля по описанию из Яндекс-хендбука, поскольку библиотека implicit поддерживают только implicit-фидбек, а у нас явные оценки (explicit). Далее подобрал гиперпараметры (число факторов, регуляризация и количество итераций) с помощью Optuna. Полученные результаты на валидационной выборке подтвердили, что ALS является сильным и устойчивым baseline-методом для задачи явных рекомендаций, хоть и занимает больше времени(очень сильно зависит от гиперпараметра «число факторов», он задает размерность скрытого латентного пространства, в котором представляем и пользователей, и айтемы; а также зависит от числа итераций), чем классический SVD.

6.1.1 Матричные разложения: Truncated SVD и ALS

Почему SVD и ALS?

- **Truncated SVD:**
  - SVD лежит в основе квантового алгоритма из статьи Kerenidis–Prakash, поэтому мне показалось логичным сравнить классический подход с его квантовой версией.

    Из преимуществ можно выделить:
  - Очень понятный и отлаженный метод: в библиотеке `sklearn` метод уже реализован, так что можно пользоваться им из коробки. Это позволяет быстро получить латентные представления, не прикладывая усилий (что я, собственно, и сделал).
  - Еще он хорошо работает CSR–матрицах и масштабируется по числу пользователей и айтемов, что отлично подходит под наш случай (для масштабирования датасета самое то)
- **Alternating Least Squares (ALS):**
  - ALS является одним из самых популярных методов для RecSys (думаю, за счет явной L2-регуляризации)
  - В отличие от NMF, написанный ALS игнорирует пропуски (нулевые элементы разреженной матрицы) и действительно улучшает метрики на валидации, поэтому перекоса к нулям нет.

– На самом деле основная причина – у меня уже был опыт использования ALS, а поскольку суть проекта не в оптимизации классических решений, то решил немного сэкономить своего времени, чтобы перераспределить его на квантовую и гибридную части.

Как упоминал выше, ALS напрямую оптимизирует по известным оценкам с L2-регуляризацией. Такой подход даёт более точные предсказания, но за это приходится платить вычислительными мощностями и временем (на решение систем линейных уравнений для каждого пользователя и айтема).

Truncated SVD (sklearn)

В основе метода лежит классическое сингулярное разложение матрицы $R \in \mathbb{R}^{m \times n}$:

$$R \approx U\, S\, V^\top, \quad U \in \mathbb{R}^{m \times k},\ S \in \mathbb{R}^{k \times k},\ V \in \mathbb{R}^{n \times k},$$

где $k \ll \min(m, n)$ (кстати, это условие фигурирует во многих статьях про матричные аппроксимации, в том числе и в Kerenidis–Prakash). В ноутбуке используется `sklearn.decomposition.TruncatedSVD`, который в качестве входных параметров принимает число компонент `n_components` — по факту ранг матрицы, которой мы будем приближать нашу исходную — и число итераций `n_iter`. В коде:

- `svd = TruncatedSVD(n_components=k, n_iter=T)`
- `U = svd.fit_transform(R_train_scaled)`
- `S = svd.singular_values_`, `V = svd.components_`
- Восстановление: $\hat{R} = U\,\mathrm{diag}(S)\,V$.

Truncated SVD хорошо работает на CSR-разреженных матрицах, быстро извлекает латентные представления пользователей и айтемов, но у него нет регуляризации. Латентные признаки извлекались в коде так (они используются для квантовых цепочек):

$$\Sigma = \sqrt{S},\ \text{user\_lat} = \mathrm{U}\Sigma,\ \text{item\_lat}^\top = \Sigma V^\top$$

Alternating Least Squares (ALS)

ALS решает задачу минимизации регуляризованной суммы квадратов ошибок по ненулевым элементам:

$$\min_{X \in \mathbb{R}^{m \times k},\, Y \in \mathbb{R}^{n \times k}} \sum_{(u,i) \in \Omega} (r_{ui} - x_u^\top y_i)^2 + \lambda \Big(\sum_u C_u \|x_u\|^2 + \sum_i C_i \|y_i\|^2 \Big),$$

где Ω — множество известных оценок, C_u, C_i — число взаимодействий пользователя u и айтема i. Алгоритм можно разбить на два формальных шага, которые появторяем до достижения сходимости:

1. При фиксированной Y для каждого пользователя u решаем систему нормальных уравнений

   $$x_u = \left(Y_{\Omega_u}^\top Y_{\Omega_u} + \lambda C_u I\right)^{-1} (Y_{\Omega_u}^\top r_u)\,,$$

   где Y_{Ω_u} — подматрица строк y_i для i в Ω_u.

2. При фиксированном X аналогично обновляем каждый вектор y_i:

   $$y_i = \left(X_{\Omega_i}^\top X_{\Omega_i} + \lambda C_i I\right)^{-1} (X_{\Omega_i}^\top r_i)\,.$$

Другими словами, «замораживаем» одну матрицу, оптимизируем вторую, а затем наоборот. По итогу получаем успешный успех через какое-то количество итераций.
В коде реализована функция `als_train(R, n_factors, reg, n_iter)`:

- Инициализация $U, X \sim \mathcal{N}(0, 1/n\_factors)$ и $V, Y \sim \mathcal{N}(0, 1/n\_factors)$.
- В цикле по n_iter обновляем поочерёдно все строки U и V
- Параметры ($n\_factors$, λ, n_iter) подбираются через Optuna, минимизируем RMSE на валидации.

Это был краткий пересказ Яндекс-хендбука. Еще повторно отмечу, что ALS очень чувствителен к параметру n_factors (число факторов) и n_iter (число итераций), потому что их увеличение значительно повышает количество затрачиваемого времени, но насколько выгодно (с точки зрения прироста качества к времени) повышать значение одного параметра при фиксированном втором — сказать не ручаюсь, эксперименты не проводил.

6.2 Квантовое SVD

6.2.1 Общая схема (Kerenidis–Prakash)

Алгоритм квантового SVD опирается на представление пользователь–объектных векторов как амплитудного состояния квантовой системы. В теории он позволяет восстановить скалярное произведение латентных векторов за полилогарифмическое время, если данные загружены в QRAM. На практике в симуляторе такой роскоши у нас нет, поэтому пришлось вопользоваться приближениями.

6.2.2 Аппроксимация без QRAM

- Вместо QRAM используется классический `TruncatedSVD` для получения матриц $U \in \mathbb{R}^{m\times k}$ и $V^\top \in \mathbb{R}^{k\times n}$.
- Сравнение двух вариантов встраивания:
  - *AmplitudeEmbedding*: загружаем весь вектор в амплитуды n кубитов;
  - *AngleEmbedding*: кодируем каждую компоненту как угол $RY(\phi_i)$ на отдельном кубите.

6.2.3 Реализация на Pennylane

- **SWAP–тест** (`swap_test`):
  - Используется устройство `default.qubit` с $1 + 2n_{\text{lat}}$ wires: 1 контрольный кубит (номер 0) и по $n_{\text{lat}} = \lceil \log_2 k \rceil$ проводов для каждого из векторов u и v.
  - На wires $1 \dots n_{\text{lat}}$ и $(1{+}n_{\text{lat}}) \dots (1{+}2n_{\text{lat}})$ последовательно выполняются два `AmplitudeEmbedding` для u и v.
  - Перед и после embedding'ов на контрольный провод (0) подаётся Hadamard. Между ними выполняются `CSWAP` по каждой паре (a, b), где $a \in [1, n_{\text{lat}}]$, $b \in [1 + n_{\text{lat}}, 1 + 2n_{\text{lat}}]$.
  - Финальный Hadamard на проводе 0, затем измеряется вероятность выхода в $|0\rangle$:
$$p_0 = \Pr(0), \quad \text{swap\_test}(u, v) = \big(p_0, 1 - p_0\big).$$
  - Для вычисления скалярного произведения используется $\text{dot\_q}(u, v) = (2p_0 - 1)\,\|u\|\|v\|$.
- **Variational Circuit с AngleEmbedding** (`var_net`):
  - Устройство с $2k$ проводами, где $k = \dim(u) = \dim(v)$.
  - `AngleEmbedding` по k проводов для $u_{\text{angles}} = u \cdot \pi$ (в диапазоне $[0, \pi]$) и ещё по k проводов для v_{angles}, оба с вращением `rotation="Y"`.
  - L слоёв параметризованных поворотов:
$$\text{для } \ell = 0 \dots L-1,\ \forall q \in [0, 2k-1]:\ RY\big(\theta_{\ell,q}\big)\,.$$
  - После каждого слоя RY применяется запутывание `layer_entangle()`, т.е. CNOT по парам (a, b), где $a \in [0, k-1]$, $b \in [k, 2k-1]$.
  - На wire 0 измеряется ожидание $\langle Z_0 \rangle$. Выход сети:
$$\hat{r} = s\ \langle Z_0 \rangle\ +\ b,$$
    где s и b — обучаемые параметры (`scale`, `bias`).
- **Обучение**:
  - Пакетная оптимизация батчей размера 16 с помощью `qml.optimize.AdamOptimizer(stepsize=0.05)`.
  - Функция потерь RMSE: $\text{rmse\_on\_batch} = \frac{1}{N}\sum(\hat{r} - r_{\text{true}})^2$.

6.2.4 Проблемы и ограничения

- Отсутствие QRAM в симуляторе вынуждает использовать классический SVD внутри эмбеддинга.
- Каждый вызов QNode имеет большой overhead на конвертацию в PennyLane и обратно.
- Нет поддержки пакетной обработки сразу N запросов: приходится вызывать квантовую функцию по одному.

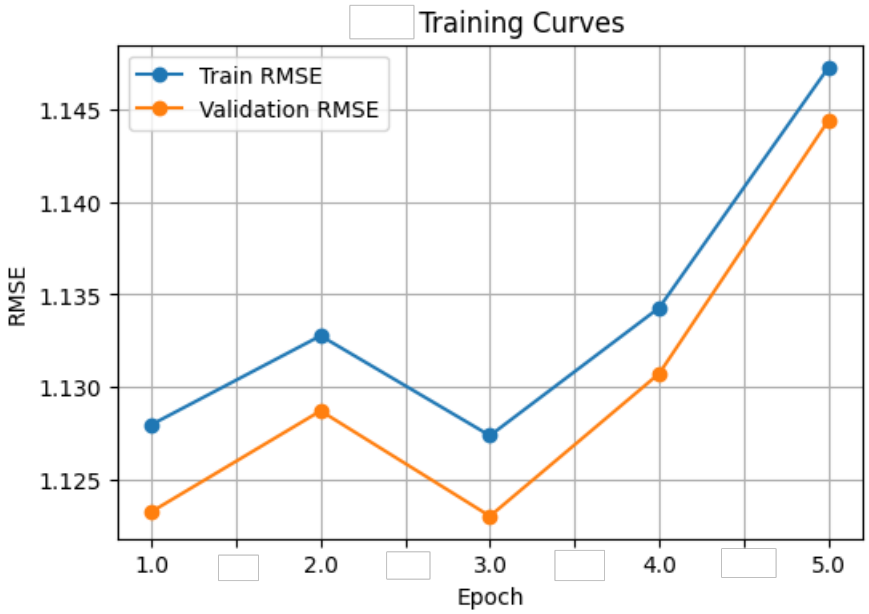

Рис. 3: Кривые обучения для var_net с Angle Ebedding. Как видим, метрика меняется незначительно (во втором знаке после запятой). При этом каждая эпоха занимает ≈ 40 минут

Такой вид графика дает еще один аргумент в поддержку к предположению, что алгоритм просто пытается выдавать среднее значение везде, а не подстраивается под конкретного пользователя.

6.3 Гибридный подход: квантовые предсказания + LSTM

6.3.1 Идея: обучать LSTM корректировать квантовые ошибки

Квантовый предиктор генерирует грубые рекомендации с RMSE. Цель — обучить LSTM подсказывать поправку $e = r_{\text{true}} - r_{\text{q}}$, чтобы итоговая оценка

$$\hat{r} = r_{\text{q}} + \hat{e}$$

имела более низкий RMSE.

6.3.2 Архитектура LSTM (QuantumErrorLSTM)

- **Входы**: непрерывный признак `quantum_pred`, ID пользователя и ID айтема.
- **Embedding**: $\text{Emb}_u(n_{\text{users}}, 32)$, $\text{Emb}_i(n_{\text{items}}, 32)$.
- **LSTM**: входной размер $1 + 32 + 32 = 65$, скрытый размер 64, 1 слой, `batch_first=True`.
- **Голова**: Dropout(0.1) + Linear(64→1) даёт предсказание ошибки $\hat{e}$.
- **Loss**: RMSE, оптимизация `Adam` LR=1e-4, ранняя остановка по RMSE на валидации.

6.3.3 Эффект: расширение диапазона предсказаний, снижение RMSE

- Диапазон квантовых предсказаний был узким: [3.40, 3.64] вокруг среднего.
- LSTM-скорректированные оценки покрывают более широкий спектр:
  имеем на train диапазон предсказанной ошибки [-3.19 ... 2.05], то есть уходим от среднего
  (хоть это и не дает сильного прироста по метрике, но качественно для пользователя стало лучше)

$$\text{RMSE}_{\text{Quantum}} \approx 1.12 \;\rightarrow\; \text{RMSE}_{\text{Hybrid}} \approx 1.09.$$

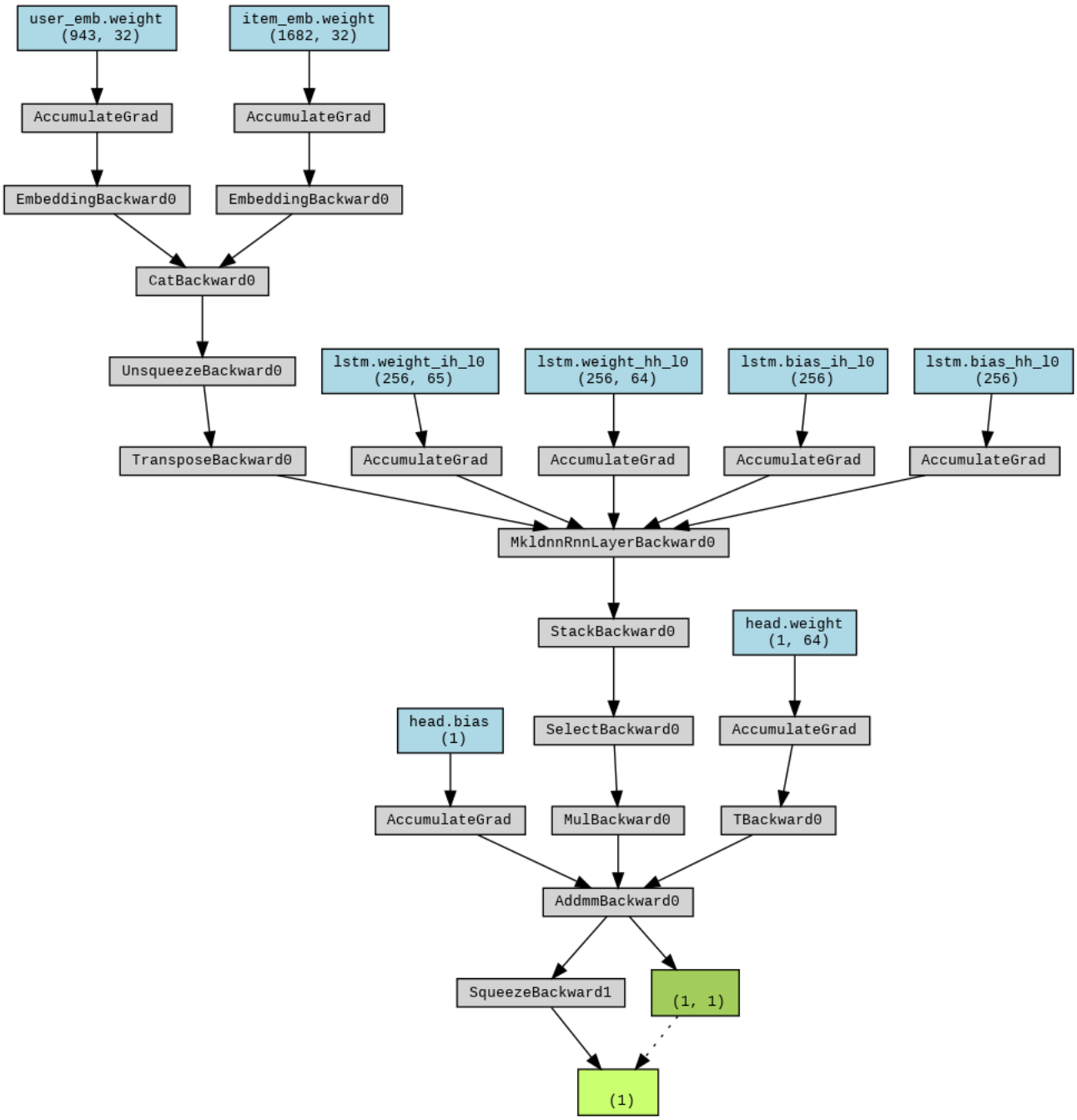

Рис. 4: Архитекутра используемой LSTM