

Low-rank self-play fine-tuning for small LLMs

П. Ю. Мун¹, Н. В. Охотников², А. В. Грабовой³

В работе исследуется проблема дообучения больших языковых моделей (LLM) в условиях ограниченных ресурсов. Под ограниченными ресурсами понимается видеопамять, человеческое участие и время обучения. В работе рассматриваются модели до 1.5B. Предлагается метод дообучения, основанный на внедрении адаптеров LoRA, малоранговых разложений матриц, в слои архитектуры трансформера, и использовании стратегии self-play - текущая итерация генерирует предсказания, а обучающаяся повышает качество с помощью разграничения настоящих предсказаний от сгенерированных. Метод может снизить количество обучаемых параметров в 10000, и память в три раза, также он не требует размеченных данных помимо используемых на этапе SFT. Для анализа качества метода будет использована группа датасетов, таких как MMLU, Winogrande.

Ключевые слова: LLM, LoRA, self-play, SFT

DOI:

1 Введение

Большие языковые модели (LLM) демонстрируют исключительные возможности в широком спектре областей, которые могут требовать специализированных знаний. Примерами таких областей могут служить: программирование [1], генерация текстов [6], обращения к базам данных [8]. Обычно процесс обучения LLM состоит из нескольких этапов: предварительного обучения, и этапов дообучения: SFT, обучение на предварительно размеченном наборе данных, и RLHF, во время которого необходим эксперт-человек, оценивающий ответы модели. Предварительное обучение требует огромных вычислительных ресурсов, поэтому часто используют публичные предобученные модели (Llama3 [4], Qwen2.5 [7]) и настраивают под целевую задачу. Но также проблема ограниченности ресурсов встречается и на этапах дообучения SFT и RLHF. Ограниченность ресурсов проявляется в недостатке видеопамати для хранения и обновления параметров модели, необходимости в размеченных данных для повышения качества и времени обучения.

В работе исследуются методы повышения эффективности обучения моделей в условиях ограниченных ресурсов. В частности предлагается метод дообучения LLM, который значительно снижает потребление видеопамати, а также убирает необходимость в прямом человеческом участии. Метод основан на двух идеях.

- Во-первых, внедрение адаптеров LoRA в слои трансформера. Предполагается сравнительно маленькая внутренняя размерность пространства параметров, и рассматриваемый метод снижает размерность пространства параметров.
- Во-вторых, механизме self-play для обучения адаптеров. Механизм состоит из последовательных игр модели со своей предыдущей версией. Предыдущая версия генерирует ответы по промптам части датасета на этапе SFT, а модель пытается различить настоящий ответ от сгенерированного. Общий метод исследуется в статье [2], а в данной работе он применяется исключительно к адаптерам LoRA.

Предложенный метод развивается в двух направлениях: снижение требований к видеопамяти и снижение человеческого участия. В обоих направлениях есть близкие альтернативы:

- Многие существующие решения по снижению требуемой памяти также применяют адаптеры LoRA с использованием других инструментов. Одним из лучших методов является QLoRA [3], который предлагает 4х битные float и квантизацию, значительно снижая необходимую память. В отличие от LoRA, которая снижает количество обучаемых параметров, QLoRA также уменьшает размер параметров модели с помощью 4х битных NormalFloat, что приводит к меньшим требованиям к видеопамяти. С другой стороны, QLoRA требует передового оборудования для применения, что может являться существенным ограничением в условиях нехватки ресурсов.
- В направлении снижения человеческого участия можно выделить методы оптимизирующие этап RLHF. Одним из таких является метод DPO [5], который снижает зависимость от человеческого оценивания результата. В то же время предложенный метод не требует размеченных данных, помимо тех, что используются на SFT.
- Конечной целью работы является исследование оправданности применения предложенного метода к маленьким LLM в условиях ограниченных ресурсов.

2 Связанные работы

TODO:

3 Постановка задачи

Будем обозначать за Θ - пространство всевозможных параметров трансформера, p_{θ} - модель, а $\theta \in \Theta$ - ее параметры. В задаче ставится ограничение на количество параметров модели, то есть $\|\Theta\| \leq K$. В качестве аргумента функция p_{θ} принимает последовательность $\mathbf{x} = [x_1, \dots, x_n]$ и возвращает последовательность $\mathbf{y} = [y_1, \dots, y_m]$, которые интересуются как промпт, ответ модели. Модель p_{θ} представима в виде условной плотности, которую можно представить как:

$$p_{\theta}(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^m p_{\theta}(y_j|\mathbf{x}, y_{<j})$$

где $y_{<j} = [y_1, \dots, y_{j-1}]$, для $j \in [2, \dots, m]$, а $p_{\theta}(y_1|x, y_{<1}) = p_{\theta}(y_1|x)$.

На этапе SFT ставится задача минимизации следующей лосс-функции:

$$L_{SFT}(\theta) = -\mathbb{E}_{\mathbf{x} \sim q(\cdot), \mathbf{y} \sim p_{data}(\cdot|\mathbf{x})} [\log p_{\theta}(\mathbf{y}|\mathbf{x})] \quad (1)$$

где \mathbf{x} - случайный элемент из распределения промптов $q(\cdot)$, \mathbf{y} - случайный элемент из распределения ответов $p_{data}(\cdot|\mathbf{x})$

4 Метод

В данной секции будет рассмотрен предложенный метод, позволяющий в условиях ограниченных ресурсов повысить метрики в сравнении с метриками на этапе SFT. Процесс дообучения начинается с определения базовой модели p_{θ_0} , где $\theta_0 \in \Theta$ - параметры базовой модели. Далее, на этапе SFT обучение проходит на размеченном тренировочном датасете $\{(\mathbf{x}, \mathbf{y})\}_{i=1}^n$ из распределения $q(\cdot)$ и $p_{data}(\cdot|\mathbf{x})$. Предложенный является итеративным, поэтому в дальнейшем будем обозначать за θ_t - параметры обучаемой модели на итерации t .

Теперь перейдем к введению понятий для обозначения методов.

4.1 Адаптеры LoRA

Адаптер LoRA представлен произведением двух матриц $A \cdot B$ и может быть встроен в любую обучаемую матрицу весов W_0 . Во время дообучения матрица весов W_0 заморожена, то есть не меняется, и обучаются только матрицы A и B , что может помочь снизить количество обучаемых параметров и соответственно требуемую память для дообучения. В конце модель использует сумму изначальной матрицы с адаптером, $W = W_0 + A \cdot B$.

В данной работе адаптеры LoRA встраиваются на всех слоях в матрицы: W_q, W_k, W_v - query/key/value матрицы блоков self-attention в архитектуре трансформера. Обозначим значения всех встраиваемых параметров на шаге t за $\Delta\theta_t$, тогда

$$\theta_t = \theta_0 + \Delta\theta_t$$

4.2 Подробное описание SPIN

В центре механизма self-play находится игра между: моделью игроком и моделью противником. Цель противника сгенерировать ответы, которые были бы неразличимы с настоящими ответами, а цель игрока уметь различать настоящие ответы от сгенерированных. В рамках дообучения противником является прошлая итерация модели, а игроком текущая итерация, которая обучается.

Рассмотрим итерацию $t+1$, противником на данной итерации является p_{θ_t} , которая по промптам \mathbf{x} генерирует ответы \mathbf{y}' , а игроком $p_{\theta_{t+1}}$. Метод состоит из двух шагов: обучение игрока, обновление противника

– В качестве целевой функции для первого шага используется:

$$f_{t+1} = \arg \max_{f \in \mathcal{F}_t} \mathbb{E}_{\mathbf{x} \sim q(\cdot), \mathbf{y} \sim p_{data}(\cdot|\mathbf{x})} [l(f(\mathbf{x}, \mathbf{y}) - f(\mathbf{x}, \mathbf{y}'))], \quad (2)$$

, где $l = \log(1 + \exp(-t))$, а \mathcal{F}_t - некоторое семейство функций.

– Для обновления противника используется следующая функция:

$$p_{\theta_{t+1}} = \arg \max_p \mathbb{E}_{\mathbf{x} \sim q(\cdot), \mathbf{y} \sim p(\cdot|\mathbf{x})} [f_{t+1}(\mathbf{x}, \mathbf{y})] - \lambda \mathbb{E}_{\mathbf{x} \sim q(\cdot)} \text{KL}(p(\cdot|\mathbf{x}) || p_{\theta_t}(\cdot|\mathbf{x})), \quad (3)$$

, где λ - коэффициент регуляризации похожести итераций, а KL - дивергенция Кульбака-Лейблера. Регуляризация добавлена с целью контроля обучаемой итерации, чтобы $p_{\theta_{t+1}}$ не сильно отличалась от p_{θ_t}

4.3 Одношаговое описание SPIN

Примечательно, что последнее выражение (3) на $p_{\theta_{t+1}}$ имеет аналитическое решение $\hat{p}_{\theta_{t+1}}$:

$$\hat{p}_{\theta_{t+1}}(\mathbf{y}|\mathbf{x}) \propto p_{\theta_t}(\mathbf{y}|\mathbf{x}) \exp(\lambda^{-1} f_{t+1}(\mathbf{x}, \mathbf{y})). \quad (4)$$

Важно упомянуть, что у такого аналитического решения может не быть представления в виде функции, параметризованной элементами из Θ , то есть $\hat{p}_{\theta_{t+1}}$ не обязательно принадлежит семейству $\{p_{\theta} | \theta \in \Theta\}$. Тогда рассмотрим класс функций для f_{t+1} :

$$\mathcal{F}_t = \left\{ \lambda \cdot \log \frac{p_{\theta}(\mathbf{y}|\mathbf{x})}{p_{\theta_t}(\mathbf{y}|\mathbf{x})} \middle| \theta \in \Theta \right\}, \quad (5)$$

Подставляя элементы полученного семейства в формулу 1, получим искомый критерий качества, используемый при обучении.

$$L_{SPIN} = \mathbb{E}_{\mathbf{x} \sim q(\cdot), \mathbf{y} \sim p_{data}(\cdot|\mathbf{x})} \left[\ell \left(\lambda \log \frac{p_{\theta}(\mathbf{y}|\mathbf{x})}{p_{\theta_t}(\mathbf{y}|\mathbf{x})} - \lambda \log \frac{p_{\theta}(\mathbf{y}'|\mathbf{x})}{p_{\theta_t}(\mathbf{y}'|\mathbf{x})} \right) \right], \quad (6)$$

Тогда правило обновление параметров имеет вид:

$$\theta_{t+1} = \arg \min_{\theta \in \Theta} L_{SPIN}(\theta, \theta_t) \quad (7)$$

4.4 Применение SPIN к адаптерам

Обозначим за $\Omega \subset \Theta$ - подпространство весов адаптеров LoRA в архитектуре трансформера. Заметим, что подставление параметров адаптеров LoRA $\Delta\theta$ сужает рассматриваемый класс функций $\{p_{\theta_0+\Delta\theta}|\Delta\theta \in \Omega\} \subset \{p_{\theta}|\theta \in \Theta\}$, или же снижает размерность пространства обучаемых параметров, но все равно может быть применим в полученному критерию качества L_{SPIN}

$$\Delta\theta_{t+1} = \arg \min_{\Delta\theta \in \Omega} L_{SPIN}(\theta_0 + \Delta\theta, \theta_0 + \Delta\theta_t) \quad (8)$$

Алгоритм 1 Self-Play Fine-Tuning с адаптерами LoRA

Дано: $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i \in \{1 \dots n\}}$: SFT датасет, T : Number of iterations.

для $t = 0, \dots, T - 1$

для $i = 1, \dots, N$

Генерация ответов моделью противником: $\mathbf{y}'_i \sim p_{\theta_0+\Delta\theta_t}(\cdot|\mathbf{x}_i)$.

$$\Delta\theta_{t+1} = \arg \min_{\Delta\theta \in \Omega} \sum_{i=1}^n \ell \left(\lambda \log \frac{p_{\theta_0+\Delta\theta}(\mathbf{y}_i|\mathbf{x}_i)}{p_{\theta_0+\Delta\theta_t}(\mathbf{y}_i|\mathbf{x}_i)} - \lambda \log \frac{p_{\theta_0+\Delta\theta}(\mathbf{y}'_i|\mathbf{x}_i)}{p_{\theta_0+\Delta\theta_t}(\mathbf{y}'_i|\mathbf{x}_i)} \right).$$

Вывод: $\Delta\theta_T$.

5 Вычислительный эксперимент

Целью данного эксперимента является подтверждение сохранения или повышению качества модели при дообучении предложенным методом. Особое внимание уделяется контролю затрачиваемых ресурсов и измерению времени обучения.

Для начало необходимо задать ограничение $\mathbf{K} = 5 \cdot 10^8$, отвечающий за ограничение на количество параметров модели $\|\Theta\| \leq \mathbf{K}$. Значение обусловлено используемым оборудованием Tesla T4 из Google Colab, память 16 GB, количество ядер 2560.

5.1 Описание базового эксперимента

В качестве предобученной модели взята трансформерная модель [qwen2.5-0.5B](#) для задачи генерации текстов, так как она достигает верхней границы на ограничение по

параметрам, а также является новой моделью, показывающей отличный результат на leaderboards.

Обучение происходит на датасете [GSM8K](#), состоящем из более чем 8000 математических задач на простую арифметику: $+$, $-$, $\%$, \cdot . Все задачи данного уровня должен быть способен решать ученик средних классов. Ответ представлен в виде рассуждения, использующего математические вычисления.

Во время обучения в матрицы: W_q, W_k, W_v были встроены адаптеры LoRA и соответственно остальные параметры были заморожены. Обучение происходило на трех эпохах, а размер батча был равен 2.

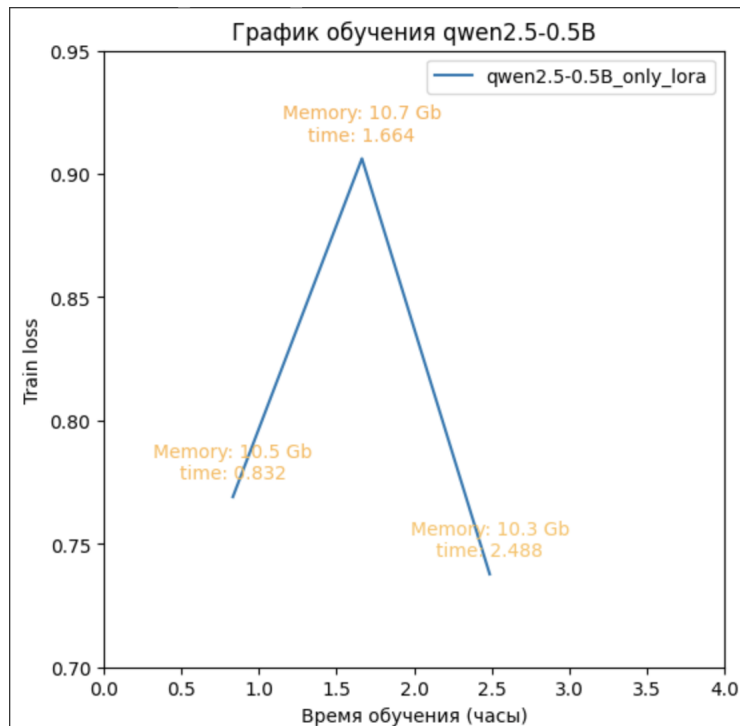
5.2 Результаты

[Так как особо результатов нет, то будет просто описан результат по единственной обученной модели]

[Ссылка на код](#)

Обучение [qwen2.5-0.5B](#) с ограничением только в 4000 объектов датасета заняло 8955 секунд ~ 2.5 часа. В процессе обучения пиковое значение используемой видеопамати достигало 10.7 GB, причем при запуске модели с размером батча равным 4 пиковое значение используемой памяти превышало максимум. Итоговый лосс на обучающей выборке составил 0.7377.

Далее будет представлен график обучения модели, так как того требует задание.



На графике изображено значение лосс функции во время обучения по эпохам

[Также вот таблица для будущей визуализации результатов]

	Model + Method	$ H $	task type 1	task type 2	...	task type m
Only LoRA	— // —					
	— // —					
LoRA + SPIV	— // —					
	— // —					

Рассматриваются столбцы: модель + примененный метод дообучения, количество па-
раметров, тип задачи (text generation, classification и т.д.) или датасеты, на котором про-
исходит дообучения (не знаю что лучше использовать). Слева изображены две графы,
они будут убраны.

6 *

Список литературы

[1] Mark Chen и др. «Evaluating Large Language Models Trained on Code». В: *CoRR* abs/2107.03374 (2021). DOI: [10.48550/arxiv.2107.03374](https://arxiv.org/abs/2107.03374). arXiv: [2107.03374](https://arxiv.org/abs/2107.03374). URL: <https://arxiv.org/abs/2107.03374>.

[2] Zixiang Chen и др. «Self-Play Fine-Tuning Converts Weak Language Models to Strong Language Models». В: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: <https://openreview.net/forum?id=04cHTxW9BS>.

[3] Tim Dettmers и др. «QLoRA: Efficient Finetuning of Quantized LLMs». В: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Под ред. Alice Oh и др. 2023. URL: http://papers.nips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html.

[4] Abhimanyu Dubey и др. «The Llama 3 Herd of Models». В: *CoRR* abs/2407.21783 (2024). DOI: [10.48550/ARXIV.2407.21783](https://arxiv.org/abs/2407.21783). arXiv: [2407.21783](https://arxiv.org/abs/2407.21783).

[5] Rafael Rafailov и др. «Direct Preference Optimization: Your Language Model is Secretly a Reward Model». В: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Под ред. Alice Oh и др. 2023. URL: http://papers.nips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

[6] Hugo Touvron и др. «Llama 2: Open Foundation and Fine-Tuned Chat Models». В: *CoRR* abs/2307.09288 (2023). DOI: [10.48550/ARXIV.2307.09288](https://arxiv.org/abs/2307.09288). arXiv: [2307.09288](https://arxiv.org/abs/2307.09288).

[7] An Yang и др. «Qwen2.5-1M Technical Report». В: *CoRR* abs/2501.15383 (2025). DOI: [10.48550/ARXIV.2501.15383](https://arxiv.org/abs/2501.15383). arXiv: [2501.15383](https://arxiv.org/abs/2501.15383).

- 176 [8] Victor Zhong, Caiming Xiong и Richard Socher. «Seq2SQL: Generating Structured Queries
177 from Natural Language using Reinforcement Learning». В: *CoRR* abs/1709.00103 (2017).
178 DOI: [10.48550/arxiv.1709.00103](https://doi.org/10.48550/arxiv.1709.00103). arXiv: [1709.00103](https://arxiv.org/abs/1709.00103). URL: [http://arxiv.org/abs/](http://arxiv.org/abs/1709.00103)
179 [1709.00103](http://arxiv.org/abs/1709.00103).