УДК 004.85

ОПТИМІЗАЦІЙНІ МЕТОДИ ДЛЯ НАВЧАННЯ ГЕНЕРАТИВНИХ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

Холоденко В.С.
Науковий керівник – к.т.н., проф. Рябова Н.В.
e-mail: vladyslav.kholodenko@nure.ua
Харківський національний університет радіоелектроніки, каф. ШІ
м. Харків, Україна

This comprehensive analysis explores advanced optimization methods for training Generative Large Language Models (LLMs) to overcome profound memory and computational bottlenecks. It reviews the evolution from foundational first-order techniques, like AdamW, to memory-efficient alternatives such as Adafactor and Lion. The text further examines second-order and preconditioned strategies (e.g., Sophia, SOAP, Muon) that adapt to landscape curvature, alongside low-rank subspace projection frameworks like GaLore that drastically reduce memory footprints. Additionally, it highlights temporal dynamics methods like MARS, Schedule-Free, and Grokfast for accelerated convergence.

Швидке масштабування генеративних великих мовних моделей (LLM) створило значні проблеми з обчисленнями та пам'яттю. Стандартний стохастичний градієнтний спуск не справляється зі складними ландшафтами функцій втрат глибоких мереж, що змушує покладатися на адаптивні градієнтні методи, апроксимації другого порядку та методи стиснення пам'яті. Головним викликом у сучасній оптимізації LLM є балансування між ефективністю використання пам'яті, обчислювальною пропускною здатністю та здатністю до узагальнення. Дана робота спрямована на пошук вирішення поставленої проблеми з урахуванням зазначених вимог.

Для вирішення цих проблем базові методи оптимізації першого порядку значно еволюціонували. AdamW залишається стандартним базовим методом, який значно покращує узагальнення шляхом відокремлення зменшення ваги (weight decay) від адаптивного масштабування швидкості навчання [1]. Однак його вимога зберігати два повнорозмірні акумулятори моментів для кожного параметра суттєво обмежує масштабованість. Щоб подолати цей бар'єр пам'яті, Adafactor досягає сублінійних витрат пам'яті, відстежуючи лише суми по рядках і стовпцях ковзних середніх квадратів градієнта, що дозволяє вмістити масивні моделі в існуючі обмеження пам'яті. З іншого боку, оптимізатор Lion використовує операцію на основі знаку для задання рівномірних напрямків оновлення. Відстежуючи лише імпульс (momentum) і відкидаючи другий момент, Lion потребує на 50% менше пам'яті для

стану оптимізатора порівняно з AdamW, хоча він залишається чутливим до розмірів пакету (batch sizes).

Оскільки методи першого порядку наближено оцінюють кривизну, оптимізатори другого порядку використовують матрицю Гессе для математично обґрунтованої адаптації до кривизни. Sophia відмовляється від повної матриці Гессе на користь стохастичної діагональної оцінки. Застосовуючи поелементне відсікання (clipping), Sophia приборкує швидкі зміни ландшафту, досягаючи двократного прискорення реального часу порівняно з AdamW [2]. Щоб вловити позадіагональні кореляції між параметрами, SOAP запускає Adam виключно в межах повільно змінного координатного базису передумовлювача (preconditioner) Shampoo. Оскільки кешування обернених матриць може спричинити розбіжність, SPlus запроваджує механізм миттєвої нормалізації знаку, структурно обмежуючи оновлення, щоб дозволити безпечне кешування передумовлювача на сотні кроків. Тим часом Muon націлений на внутрішні параметри 2D-матриці, використовуючи високоефективні для апаратного забезпечення ітерації Ньютона-Шульца, а COSMOS розділяє градієнтні матриці, щоб застосувати SOAP до критичних власних підпросторів, а Muon – до решти.

Іншою надзвичайно успішною парадигмою є низькорангове проєктування та навчання в підпросторі. GaLore оминає вузькі місця пам'яті, проєктуючи багатовимірну матрицю градієнтів у компактну низькорангову форму за допомогою сингулярного розкладу матриці (SVD) [3].

Стани оптимізації обчислюються в цьому стиснутому просторі, зменшуючи пам'ять оптимізатора до 82,5% (при 8-бітному квантуванні) і дозволяючи попереднє навчання LLM з 7 млрд параметрів на одній 24 ГБ відеокарті. Спираючись на це, Natural GaLore інтегрує природний градієнтний спуск другого порядку безпосередньо в низькоранговий фреймворк, використовуючи матричну тотожність Вудбері, досягаючи швидкості методів другого порядку практично без додаткових витрат пам'яті.

Маніпулювання часовою послідовністю градієнтів додатково прискорює збіжність. Оптимізатор MARS об'єднує зменшення дисперсії з адаптивним передумовленням за допомогою методу масштабованого стохастичного рекурсивного імпульсу для придушення шуму міні-пакетів. Підхід Schedule-Free усуває заздалегідь визначені графіки швидкості навчання шляхом усереднення ітерацій, безперервно відстежуючи теоретичну межу Парето для втрат відносно часу обчислень [4]. Крім того, Grokfast застосовує фільтр низьких частот до градієнтів, посилюючи повільні низькочастотні сигнали, щоб змусити мережу надавати пріоритет виділенню правил замість запам'ятовування, прискорюючи ефект "гроккінгу" (grokking). На системному рівні оптимізатор нульової

надлишковості (ZeRO) слугує фізичним рушієм для цих алгоритмів, розділяючи стани оптимізатора, градієнти та параметри моделі між розподіленими графічними процесорами для лінійного масштабування зменшення обсягу пам'яті.

Ґрунтуючись на цьому аналізі, пропоноване дослідження має за мету вирішення проблеми компромісу між надмірним споживанням пам'яті (через стани оптимізатора) та швидкістю збіжності при навчанні масивних LLM на доступному обладнанні. Працюючи на перетині навчання в підпросторі та методів другого порядку, пропонується новий гібридний фреймворк, що поєднує низькорангові проєкції GaLore зі стохастичною оцінкою матриці Гессе від Sophia.

Передбачається, що обчислення цих оцінок виключно у стиснутому підпросторі дозволить значно зменшити витрати пам'яті на оптимізацію другого порядку.

Крім того, для усунення негнучкості жорстко заданої тривалості навчання планується інтегрувати механізм усереднення ітерацій Schedule-Free у цю архітектуру. Головне завдання – адаптувати математику усереднення для роботи у спроєктованому підпросторі, повністю позбувшись потреби у вгадуванні графіків швидкості навчання. Очікується, що ця єдина методологія значно знизить вимоги до VRAM, зберігши або перевершивши швидкість збіжності сучасних методів, що дозволить розширити межу Парето для навчання LLM на споживчому обладнанні.

Список використаних джерел:

1. Optimizing Large Language Models: A Deep Dive into Effective Prompt Engineering Techniques – MDPI, 2026, URL: https://www.mdpi.com/2076-3417/15/3/1430 (дата звернення: 11.03.2026).

2. Frontiers in Artificial Intelligence Algorithm Optimization: A Comprehensive Review of Training, 2026, URL: https://journals.zeuspress.org/index.php/CAI/article/download/322/281/1133 (дата звернення: 11.03.2026).

3. Sophia: A Scalable Stochastic Second-order Optimizer for Language Model Pre-training – arXiv, 2026, URL: https://arxiv.org/abs/2305.14342 (дата звернення: 11.03.2026).

4. MARS: Unleashing the Power of Variance Reduction for Training Large Models – arXiv, 2026, URL: https://arxiv.org/abs/2411.10438 (дата звернення: 11.03.2026).