GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection

разбор статьи
https://arxiv.org/abs/2403.03507

Дмитрий Колодезев @promsoft kolodezev.ru
2024.03.19 @ DataTalk

GaLore: Memory-Efficient LLM Training

Jiawei Zhao [1] **Zhenyu Zhang** [3] **Beidi Chen** [2 4] **Zhangyang Wang** [3] **Anima Anandkumar** [* 1] **Yuandong Tian** [* 2]

Abstract

Training Large Language Models (LLMs) presents significant memory challenges, predominantly due to the growing size of weights and optimizer states. Common memory-reduction approaches, such as low-rank adaptation (LoRA), add a trainable low-rank matrix to the frozen pre-trained weight in each layer, reducing trainable parameters and optimizer states. However, such approaches typically underperform training with full-rank weights in both pre-training and fine-tuning stages since

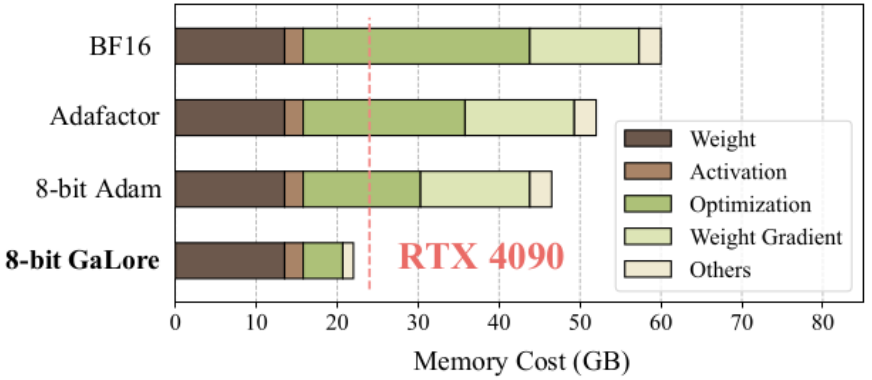


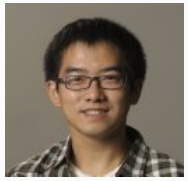
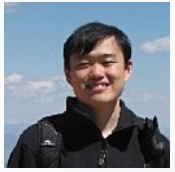
Figure 1: Memory consumption of pre-training a LLaMA 7B model with a token batch size of 256 on a single device, without activation checkpointing and memory offloading. Details refer to Section 5.5.

Авторы

- Jiawei Zhao
- Zhenyu Zhang
- Beidi Chen
- Zhangyang Wang
- Anima Anandkumar
- Yuandong Tian





Статистика цитирования	27350	26229
h-индекс	76	74
i10-индекс	239	233

А быстро тут

- 6 марта 2024 статья на Архиве
- 7 марта 2024 репозиторий и pip
- 11 марта 2024 PR в transformers (open)
- 11 марта 2024 Релиз SWIFT ModelScope
- 13 марта 2024 Релиз LLamaFactory

- Когда у вас дойдут руки попробовать, будет везде

Какую проблему решали

- LLaMA 7B, BF16
  - 14GB параметры
  - 14x3 = 42GB Adam (1 момент + 2 момент + градиент)
  - 2GB активации
- 58 GB
- Калькулятор https://huggingface.co/spaces/Vokturz/can-it-run-llm

Какие есть варианты

- PEFT
  - LoRA — доучиваем адаптер
  - ReLoRA — учим полностью
- Проблемы
  - Хуже, чем полноранговое обучение
  - Требует полнорангового прогрева для предобучения
- Гипотезы о причинах проблем
  - Оптимальная матрица весов не-низкоранговая
  - Репараметризация меняет динамику обучения

GaLore

- Гипотеза: структура градиента
  - низкоранговая
  - медленно меняющаяся
- Давайте строить проекцию раз в 200 итераций
  - torch.linalg.svd(matrix, full_matrices = False)
- Давайте использовать оптимизатор для проекции
- А веса оставим полноранговые

Алгоритм

Algorithm 1: GaLore, PyTorch-like

```
for weight in model.parameters():
    grad = weight.grad
    # original space -> compact space
    lor_grad = project(grad)
    # update by Adam, Adafactor, etc.
    lor_update = update(lor_grad)
    # compact space -> original space
    update = project_back(lor_update)
    weight.data += update
```

GaLoRE + Adam

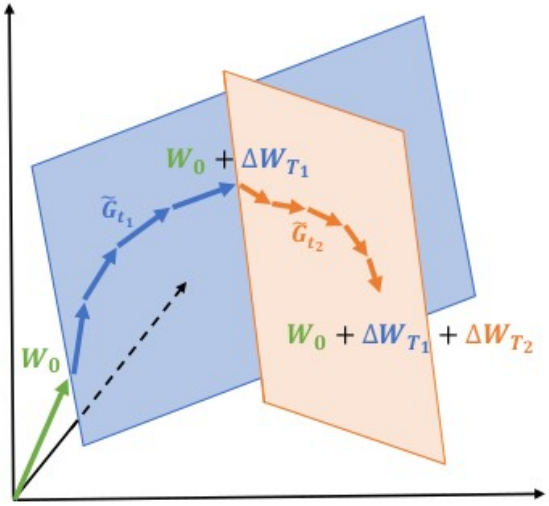


Figure 2: Learning through low-rank subspaces ΔW_{T_1} and ΔW_{T_2} using GaLore. For $t_1 \in [0, T_1 - 1]$, W are updated by projected gradients $\tilde{G}_{t_1}$ in a subspace determined by fixed P_{t_1} and Q_{t_1}. After T_1 steps, the subspace is changed by re-computing P_{t_2} and Q_{t_2} for $t_2 \in [T_1, T_2 - 1]$, and the process repeats until convergence.

Algorithm 2: Adam with GaLore

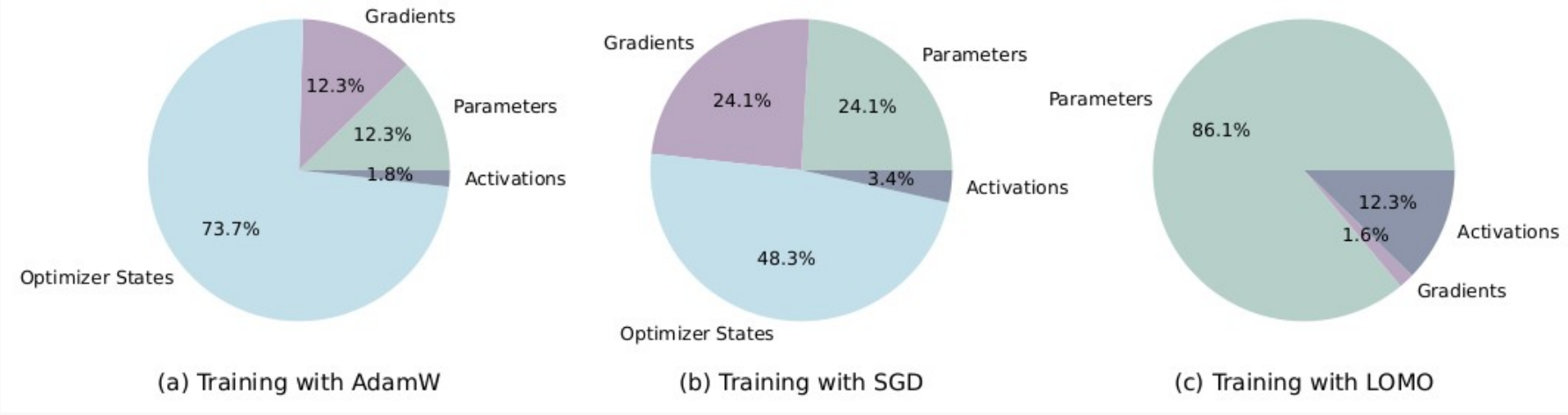
Input: A layer weight matrix $W \in \mathbb{R}^{m \times n}$ with $m \leq n$. Step size η, scale factor α, decay rates β_1, β_2, rank r, subspace change frequency T.
Initialize first-order moment $M_0 \in \mathbb{R}^{n \times r} \leftarrow 0$
Initialize second-order moment $V_0 \in \mathbb{R}^{n \times r} \leftarrow 0$
Initialize step $t \leftarrow 0$
repeat
  $G_t \in \mathbb{R}^{m \times n} \leftarrow -\nabla_W \varphi_t(W_t)$
  if $t \bmod T = 0$ **then**
    $U, S, V \leftarrow \text{SVD}(G_t)$
    $P_t \leftarrow U[:, :r]$ {Initialize left projector as $m \leq n$}
  else
    $P_t \leftarrow P_{t-1}$ {Reuse the previous projector}
  end if
  $R_t \leftarrow P_t^\top G_t$ {Project gradient into compact space}

  UPDATE(R_t) **by Adam**
    $M_t \leftarrow \beta_1 \cdot M_{t-1} + (1 - \beta_1) \cdot R_t$
    $V_t \leftarrow \beta_2 \cdot V_{t-1} + (1 - \beta_2) \cdot R_t^2$
    $M_t \leftarrow M_t / (1 - \beta_1^t)$
    $V_t \leftarrow V_t / (1 - \beta_2^t)$
    $N_t \leftarrow M_t / (\sqrt{V_t} + \epsilon)$

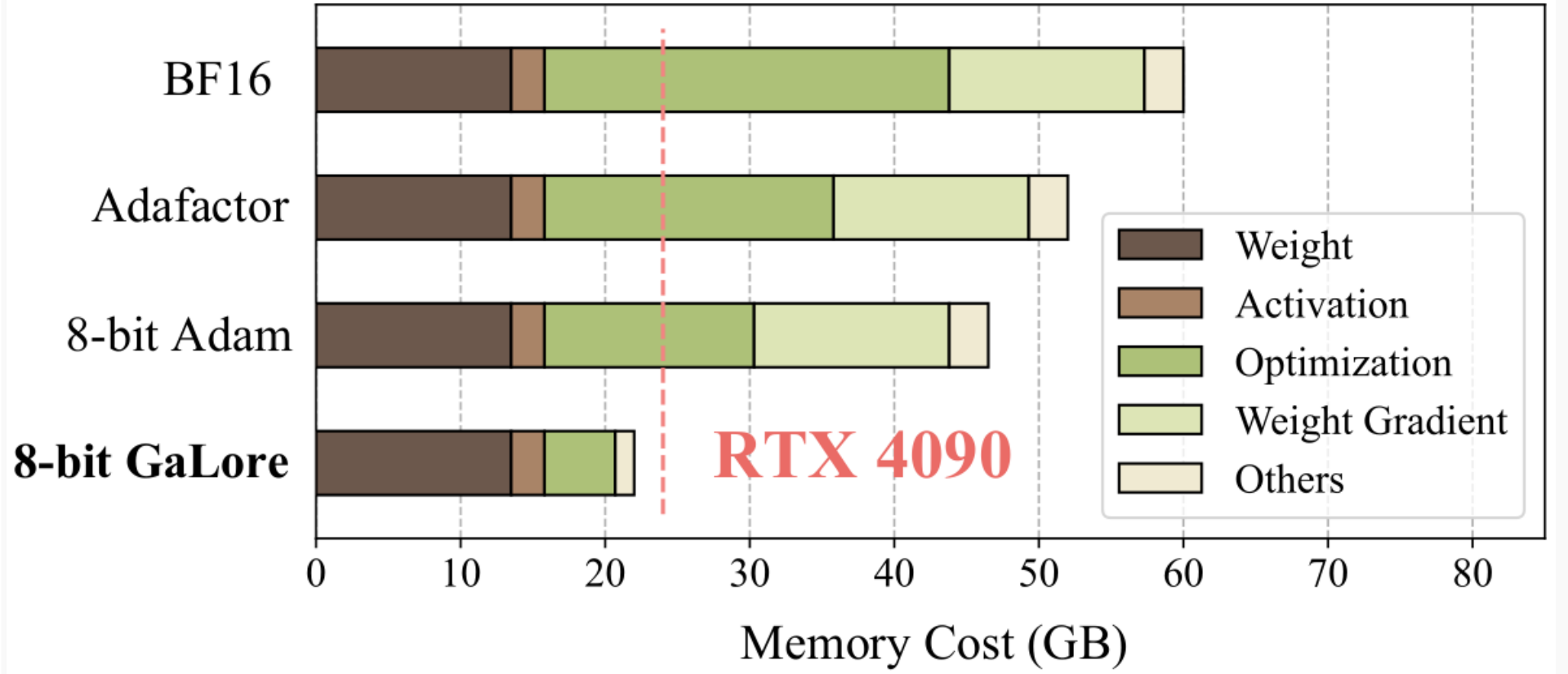
  $\tilde{G}_t \leftarrow \alpha \cdot P N_t$ {Project back to original space}
  $W_t \leftarrow W_{t-1} + \eta \cdot \tilde{G}_t$
  $t \leftarrow t + 1$
until convergence criteria met
return W_t

Что еще можно

- 8 бит
- Adafactor
- SGD гладко, без локальных оптимумов и сёдел
- Послойное применение градиента LOMO



Главная картинка



Про качество

Pre-training LLaMA 7B on C4 dataset for 150K steps

	60M	**130M**	**350M**	**1B**
Full-Rank	34.06 (0.36G)	25.08 (0.76G)	18.80 (2.06G)	15.56 (7.80G)
GaLore	**34.88** (0.24G)	**25.36** (0.52G)	**18.95** (1.22G)	**15.64** (4.38G)
Low-Rank	78.18 (0.26G)	45.51 (0.54G)	37.41 (1.08G)	142.53 (3.57G)
LoRA	34.99 (0.36G)	33.92 (0.80G)	25.58 (1.76G)	19.21 (6.17G)
ReLoRA	37.04 (0.36G)	29.37 (0.80G)	29.08 (1.76G)	18.33 (6.17G)
r/d_{model}	128 / 256	256 / 768	256 / 1024	512 / 2048
Training Tokens	1.1B	2.2B	6.4B	13.1B

Интересные 8 бит

Table 3: Pre-training LLaMA 7B on C4 dataset for 150K steps. Validation perplexity and memory estimate are reported.

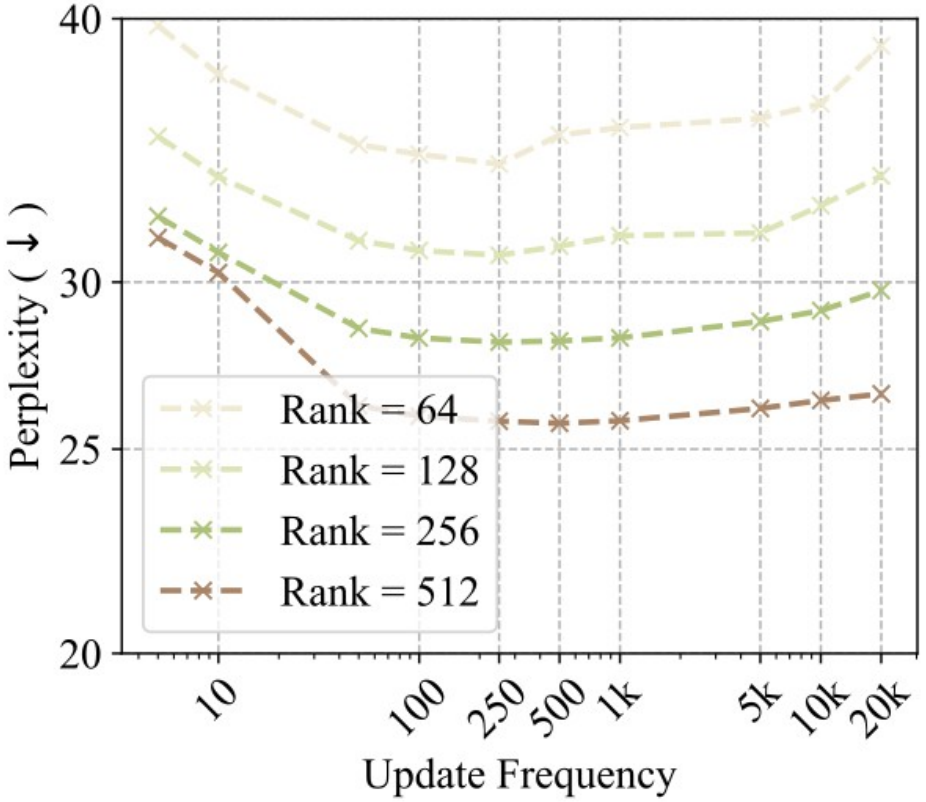
	Mem	**40K**	**80K**	**120K**	**150K**
8-bit GaLore	18G	17.94	15.39	14.95	14.65
8-bit Adam	26G	18.09	15.47	14.83	14.61
Tokens (B)		5.2	10.5	15.7	19.7

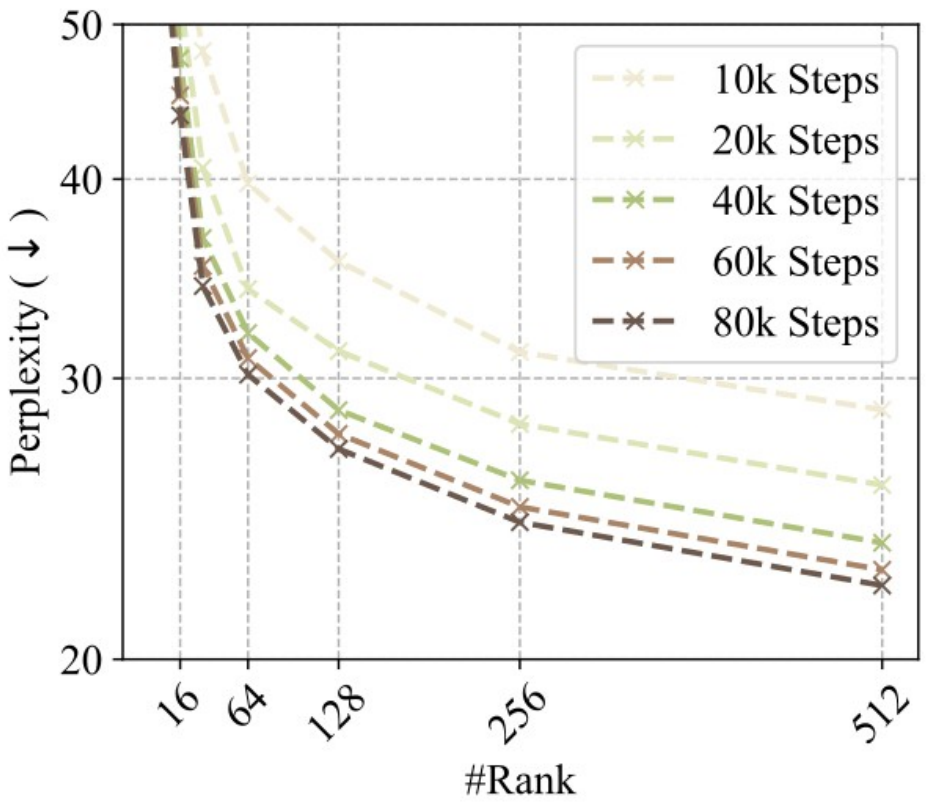
Чуть помедленнее

Table 8: Measuring memory and throughput on LLaMA 1B model.

Model Size	Layer Wise	Methods	Token Batch Size	Memory Cost	Throughput	
					#Tokens / s	#Samples / s
1B	✘	AdamW	256	13.60	1256.98	6.33
		Adafactor	256	13.15	581.02	2.92
		Adam8bit	256	9.54	1569.89	7.90
		8-bit GaLore	256	7.95	1109.38	5.59
1B	✔	AdamW	256	9.63	1354.37	6.81
		Adafactor	256	10.32	613.90	3.09
		Adam8bit	256	6.93	1205.31	6.07
		8-bit GaLore	256	5.63	1019.63	5.13

Переключаем пространства





Итого, сколько памяти

- Full 4mn
- LoRA mn+3mr +3nr
- GaLoRE mn+mr+2nr
- И еще послойно

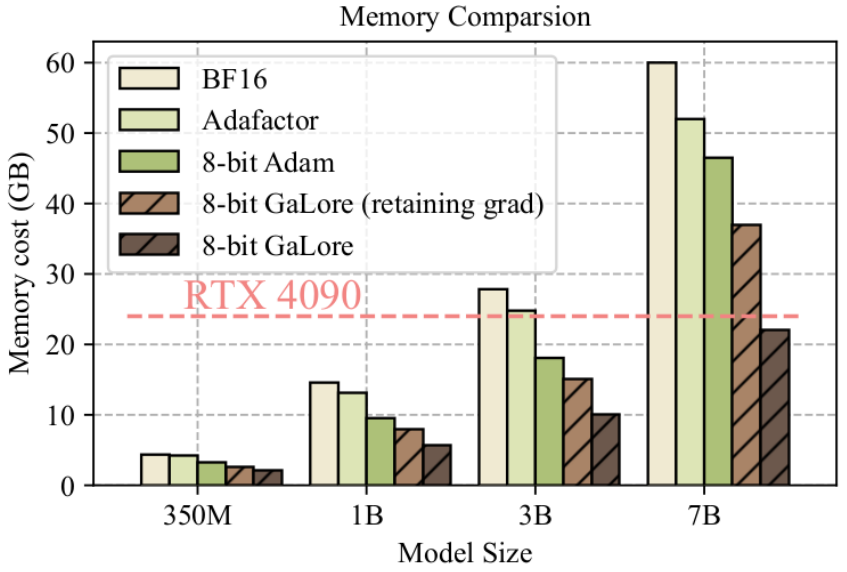


Figure 4: Memory usage for different methods at various model sizes, evaluated with a token batch size of 256. 8-bit GaLore (retaining grad) disables per-layer weight updates but stores weight gradients during training.

Разные оптимизаторы

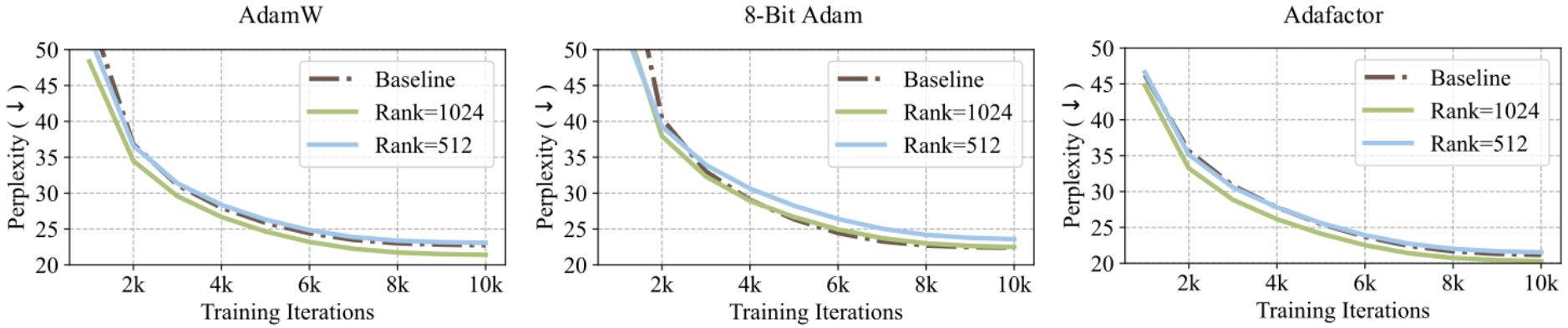


Figure 3: Applying GaLore to different optimizers for pre-training LLaMA 1B on C4 dataset for 10K steps. Validation perplexity over training steps is reported. We apply GaLore to each optimizer with the rank of 512 and 1024, where the 1B model dimension is 2048.

Гиперпараметры

- T ~ 200
- r
- α ≥1

Table 7: Hyperparameters of fine-tuning RoBERTa base for GaLore.

	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
Batch Size	16	16	16	32	16	16	16	16
# Epochs	30	30	30	30	30	30	30	30
Learning Rate	1E-05	1E-05	3E-05	3E-05	1E-05	1E-05	1E-05	1E-05
Rank Config.				$r = 4$				
GaLore α				4				
Max Seq. Len.				512				

	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
Batch Size	16	16	16	32	16	16	16	16
# Epochs	30	30	30	30	30	30	30	30
Learning Rate	1E-05	2E-05	2E-05	1E-05	1E-05	2E-05	2E-05	3E-05
Rank Config.				$r = 8$				
GaLore α				2				
Max Seq. Len.				512				

Почему так можно

- В приложении доказательства:
  - Весьма вероятно, что ранг низкий
  - Обратимо
  - Сходится
- Выглядит правдоподобно, я не математик
- Отсебятина: хорошее многообразие ЕЯ

Почитать

- GaLoRE https://arxiv.org/abs/2403.03507
- LOMO https://arxiv.org/abs/2306.09782
- GaloreAdamW
- GaloreProjector