ReLoRA: High-Rank Training Through Low-Rank Updates

Чуканов Тимофей Вячеславович, БПМИ201

11.03.2024

LoRA (Low-Rank Adaptation) recap

- **Используется для fine-tuning моделей.**
- **Заморозим всю модель, будем обучать “добавку ” ΔW к матрицам весов, которая будет иметь низкий ранг:**

$$h = (W + \Delta W)x = (W + BA)x,$$

$$W \in \mathcal{R}^{d\times k}, A \in \mathcal{R}^{r\times k}, B \in \mathcal{R}^{d\times r}, r \ll \min(d,k).$$

- **Обучаются A и B. В архитектуре сети ничего не меняется.**

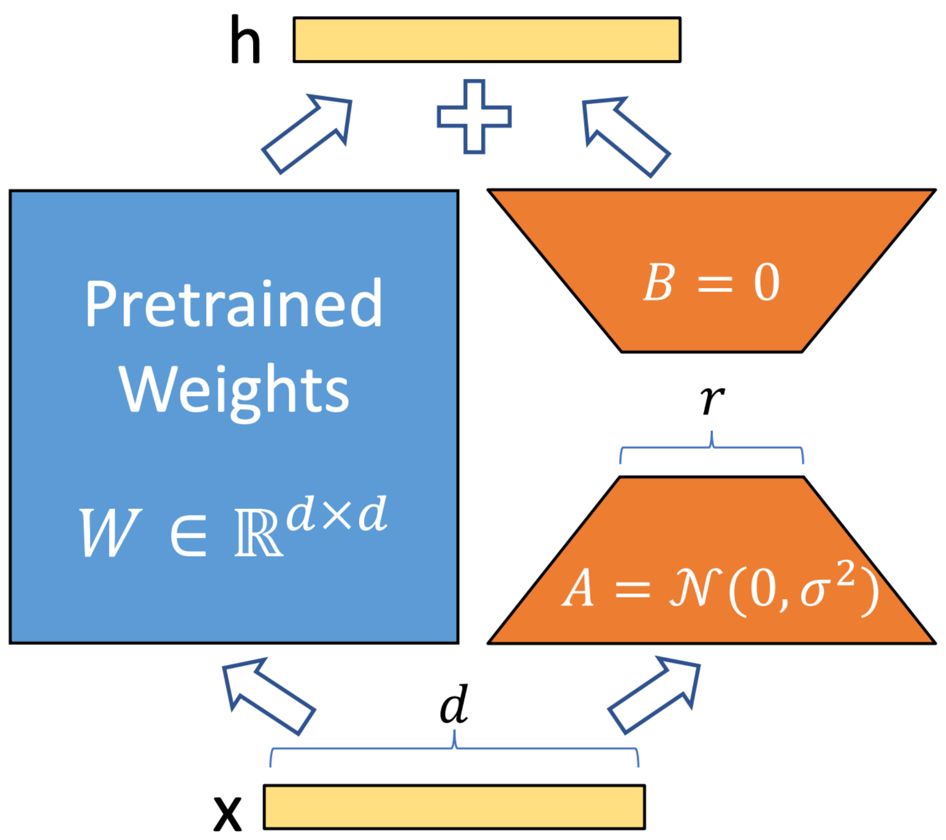

Рис. 1: устройство LoRA для одного слоя.

ReLoRA: One Simple Idea

- **В LoRA мы один раз обучали матрицы B и A, затем прибавляли их произведение к W и заканчивали.**
- **Вместо этого, будем раз в несколько шагов делать реинициализацию A и B и обучать их. А именно, сначала обучаем B_1A_1, потом перемножаем их и прибавляем к W, замораживаем $W + B_1A_1$ инициализируем новые B_2A_2 и обучаем их:**

$$\Delta W = B_1A_1 + B_2A_2 + B_3A_3 + B_4A_4 + \ldots + B_NA_N;\ h = (W + \Delta W)x$$
$$W \in \mathcal{R}^{d\times k}, A \in \mathcal{R}^{r\times k}, B \in \mathcal{R}^{d\times r}, r \ll \min(d,k).$$

- **Это позволяет использовать метод для полноценного обучения, а не только для fine-tuning.**

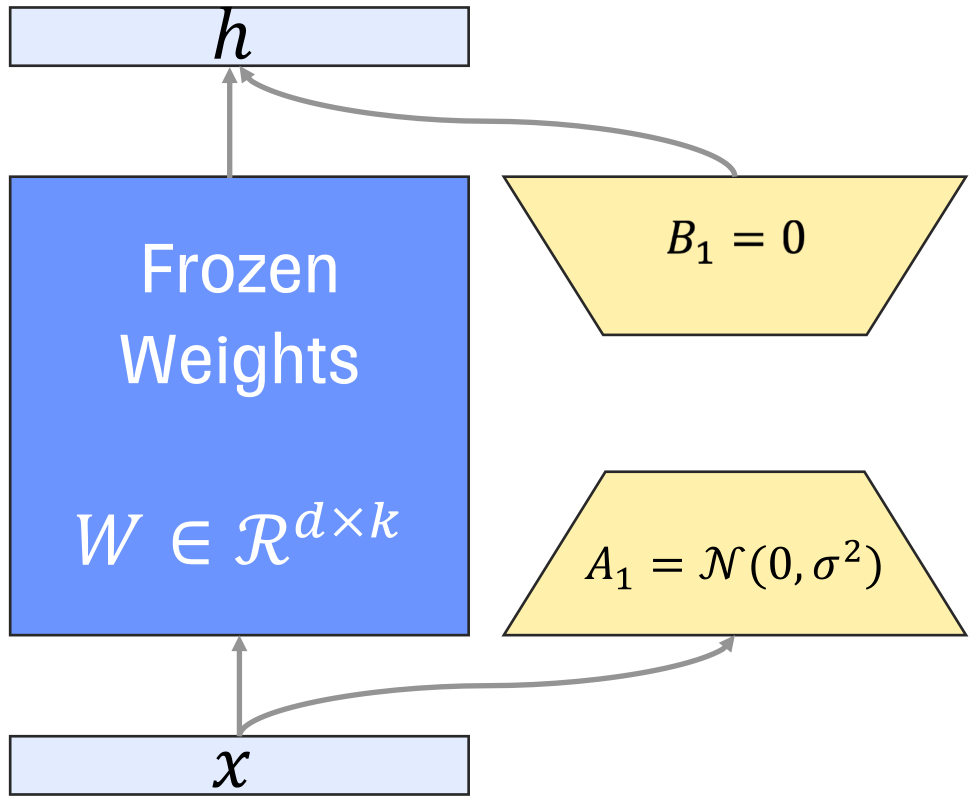

Рис. 2: шаг 1, обучаем B_1, A_1.

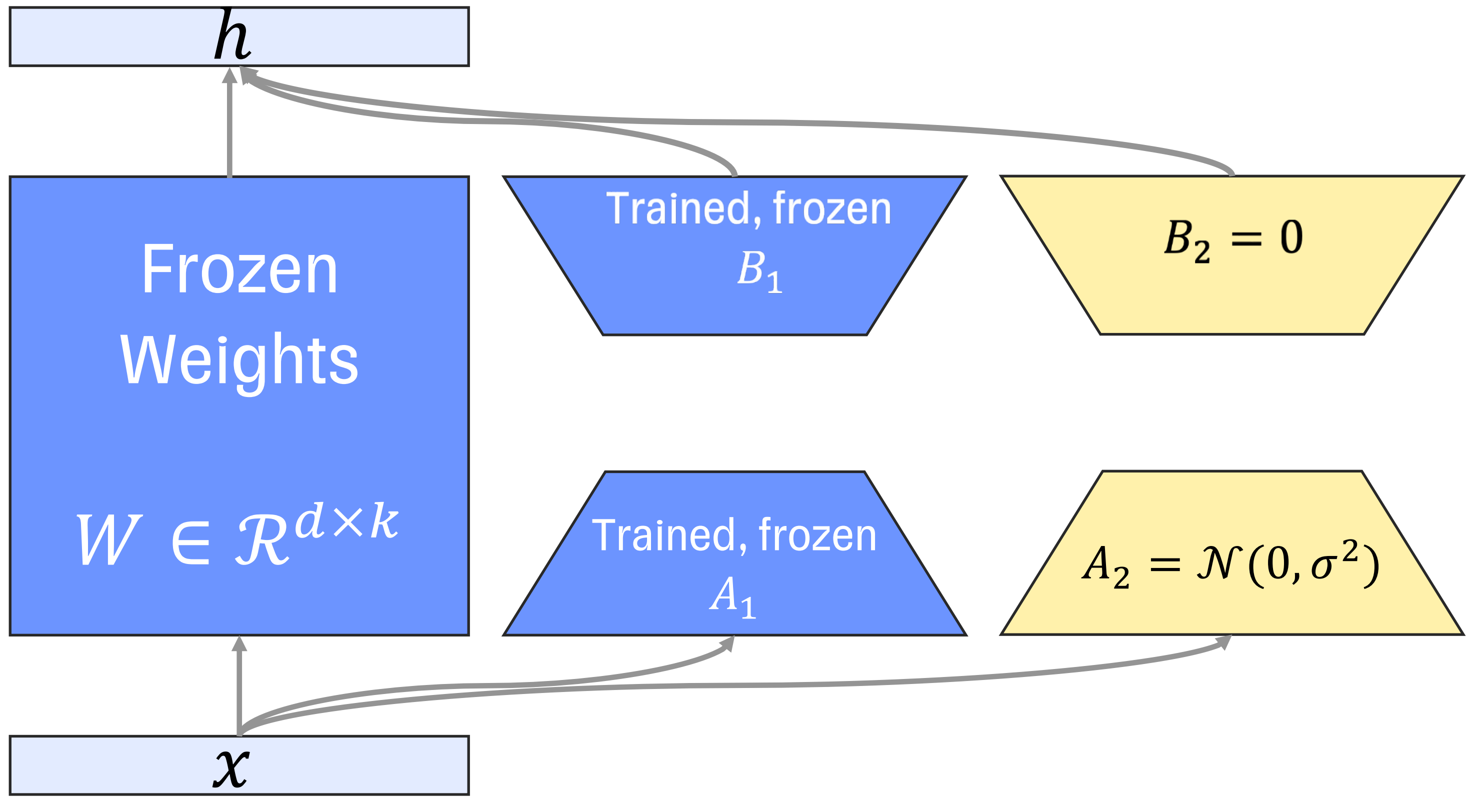

Рис. 3: шаг 2, прибавили B_1A_1 к W, заморозили $W + B_1A_1$ и обучаем B_2A_2.

ReLoRA: Sample code

Algorithm 1 ReLoRA. θ is model parameters, $\hat{\theta}$ is model parameters with linear layers replaced with ReLoRA, M and V are Adam optimizer states, η is learning rate, and q is the reinit frequency.

Require: θ, M, V, q, η

for t **in** warm start steps **do**
    Update θ, M, V, η {Regular training for warm start}
end for
for layer in model layers **do**
    if layer **is** linear **then**
        layer $\leftarrow$ ReLoRA(W^i, W^i_A, W^i_B)
        Freeze W^i
    end if
end for
for t in training steps **do**
    Update $\hat{\theta}$, M, V {Training step with ReLoRA}
    if MOD$(t, q) = 0$ **then**
        for l in model layers **do**
            if l **is** linear **then**
                $W^i \leftarrow (W^i + sW^i_A W^i_B)$
                $W^i_A \leftarrow$ kaiming_init(W^i_A); $W^i_B \leftarrow 0$
                $M_{W^i_A} \leftarrow \text{prune}(M_{W^i_A})$; $V_{W^i_A} \leftarrow \text{prune}(V_{W^i_A})$
            end if
        end for
        Start η warmup
    end if
end for
return θ

ReLoRA: Details

- **Warm Start: Перед заморозкой весов W сделаем несколько итераций полноценного обучения.**
- **Optimizer Reset: Будем делать optimizer pruning, чтобы не было сильной инерции в сторону оптимизации предыдущей пары A, B.**
- **Jagged Schedule: Чтобы обучение не развалилось после optimizer pruning, будем делать warm-up в несколько шагов learning rate с 0.**

Restarts	Optimizer Reset	Jagged Schedule	Warm Start	Perplexity (↓)
×	×	×	×	34.17
✓	×	×	×	34.25
✓	✓	×	×	*(diverged)*
✓	×	✓	×	34.29
✓	✓	✓	×	29.77
×	×	×	✓	25.46
✓	✓	✓	✓	25.04
Regular training				23.65

Table 1: Ablation studies of ReLoRA.

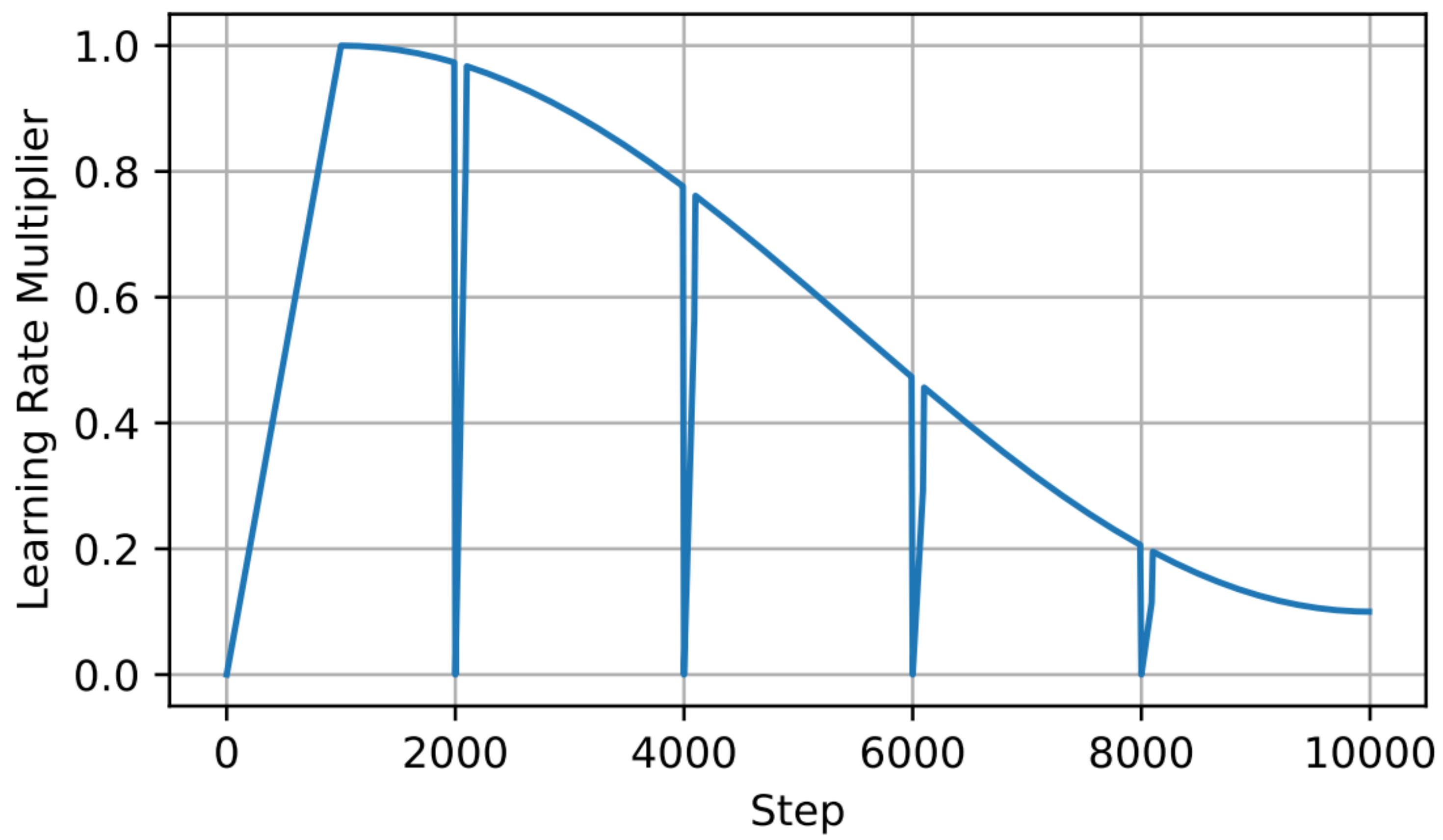

Рис. 4: Jagged cosine scheduler used in ReLoRA.

ReLoRA: Code

Algorithm 1 ReLoRA. θ is model parameters, $\hat{\theta}$ is model parameters with linear layers replaced with ReLoRA, M and V are Adam optimizer states, η is learning rate, and q is the reinit frequency.

Require: θ, M, V, q, η

for t **in** warm start steps **do**
    Update θ, M, V, η {Regular training for warm start}
end for
for layer in model layers **do**
    if layer **is** linear **then**
        layer $\leftarrow$ ReLoRA(W^i, W^i_A, W^i_B)
        Freeze W^i
    end if
end for
for t in training steps **do**
    Update $\hat{\theta}$, M, V {Training step with ReLoRA}
    if MOD$(t, q) = 0$ **then**
        for l in model layers **do**
            if l **is** linear **then**
                $W^i \leftarrow (W^i + sW^i_A W^i_B)$
                $W^i_A \leftarrow$ kaiming_init(W^i_A); $W^i_B \leftarrow 0$
                $M_{W^i_A} \leftarrow$ prune$(M_{W^i_A})$; $V_{W^i_A} \leftarrow$ prune$(V_{W^i_A})$
            end if
        end for
        Start η warmup
    end if
end for
return θ

ReLoRA Experiments. Full training

	60M	130M	250M	350M	1.3B
Full training	33.81 (60M)	23.65 (130M)	22.39 (250M)	18.66 (350M)	16.83 (250M)
Control	36.52 (43M)	27.30 (72M)	25.43 (99M)	23.65 (130M)	21.73 (250M)
LoRA	47.44 (43M)	34.17 (72M)	36.60 (99M)	57.11 (125M)	-
LoRA + Warm Start	34.73 (43M)	25.46 (72M)	22.86 (99M)	19.73 (125M)	18.23 (250M)
ReLoRA	**34.46** (43M)	**25.04** (72M)	**22.48** (99M)	**19.32** (125M)	**17.27** (250M)
Training tokens	1.2B	2.6B	6.8B	6.8B	23.1B

Table 2: Language model perplexity when trained using each of the above methods. Number of trainable parameters for each model in (brackets). Control baseline is full-rank training a model with the same total number of parameters as the number of trainable parameters in low-rank training.

	1.3B @15K steps	1.3B @20K steps	1.3B @30K steps
Full training	17.67 (250M)	17.00 (250M)	16.83 (250M)
Control	22.67 (250M)	22.00 (250M)	21.73 (250M)
LoRA + Warm Start	**18.50** (250M)	**18.38** (250M)	**18.23** (250M)
ReLoRA	**17.94** (250M)	**17.64** (250M)	**17.27** (250M)
Training tokens (billions)	11.8	15.7	23.1

Table 3: Results at 1.3B scale. Number of trainable parameters for each model in (brackets).

ReLoRA Experiments. Full training

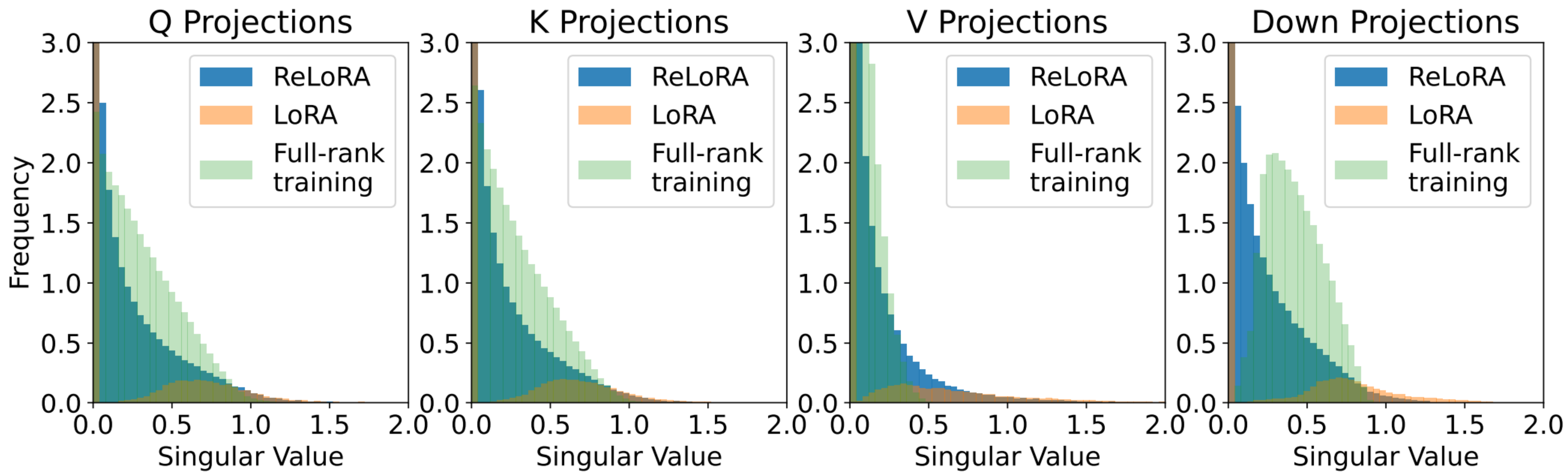

Рис. 5: Singular values spectra of the weight difference between ReLoRA and LoRA at 5,000 iterations (warm start) and 20,000 iterations. ReLoRA exhibits a closer resemblance to full-rank training than to LoRA, indicating its effectiveness in approximating full-rank behavior. 350M models.

	8xA100	**6xA6000 (Ada)**	**2x3090**
Full-rank throughput	137 ex/sec	84 ex/sec	8.8 ex/sec
ReLoRA throughput	157 ex/sec	124 ex/sec	17.8 ex/sec
Immediate speedup	15%	48%	102%
Warm-start adjusted ReLoRA throughput	149 ex/sec	111 ex/sec	14.8 ex/sec
Total speedup	9%	32%	51%

Table 4: Performance metrics in different hardware configurations. Warm start adjustment assumes 33% of full-rank training before switching to ReLoRA.

ReLoRA Experiments. Fine-tuning

Method	SST-2	MNLI	QNLI	QQP	RTE	STS-B	MRPC	CoLA	Avg
Adapters†	94.2	86.4	93.1	88.9	75.1	91.1	88.9	64.4	85.3
Prompt Tuning†	90.3	82.5	92.5	88.5	59.5	90.1	74.6	0.0	72.2
Ladder Side Tuning†	94.1	85.6	93.3	88.8	71.9	90.7	90.4	58.1	84.1
Compacter*	93.9	86.1	92.9	90.4	76.3	91.0	91.5	64.4	85.8
KronA*	94.3	86.3	93.2	90.6	77.7	91.3	92.5	63.3	86.1
Full fine-tuning*	93.6	86.2	92.8	91.7	74.8	90.1	92.7	63.4	85.7
LoRA	93.92	86.12	91.95	90.62	78.34	89.96	90.52	60.04	85.18
ReLoRA	94.15	85.96	91.68	87.2	77.74	89.88	90.03	59.92	84.57
Full fine-tuning (T5-L)	94.7	89.1	91.6	89.9	78.9	90.6	88.9	57.0	85.0
LoRA (T5-L)	95.59	89.44	93.98	91.44	85.92	90.89	92.90	63.77	87.99
ReLoRA (T5-L)	95.7	89.06	93.68	91.04	84.72	90.53	90.57	61.72	87.47

Table 5: ReLoRA for fine-tuning does not outperform LoRA. GLUE benchmark. T5-base (220M) and T5-large (770M).

Итоги

- ReLoRA – простая модификация обычного обучения, позволяющая ускорить обучение за счет небольшого снижения качества.
- Обучение с ReLoRA дает лучшие результаты, чем обычное обучение модели с таким же количеством параметров
- ReLoRA хорошо подходит для качественного обучения с ограниченным бюджетом/временем.
- ReLoRA не подходит для fine-tuning предобученных моделей, уступая в качестве многим методам, таким как LoRA, Adapters.

Lialin V. et al. ReLoRA: High-Rank Training Through Low-Rank Updates
arXiv:2307.05695

Hu E. J. et al. Lora: Low-rank adaptation of large language models
arXiv:2106.09685