Stack More Layers Differently: High-Rank Training Through Low-Rank Updates

Vladislav Lialin[*], **Namrata Shivagunde**, **Sherin Muckatira**, **and Anna Rumshisky**
University of Massachusetts Lowell

Abstract

Despite the dominance and effectiveness of scaling, resulting in large networks with hundreds of billions of parameters, the necessity to train overparametrized models remains poorly understood, and alternative approaches do not necessarily make it cheaper to train high-performance models. In this paper, we explore low-rank training techniques as an alternative approach to training large neural networks. We introduce a novel method called ReLoRA, which utilizes low-rank updates to train high-rank networks. We apply ReLoRA to pre-training transformer language models with up to 350M parameters and demonstrate comparable performance to regular neural network training. Furthermore, we observe that the efficiency of ReLoRA increases with model size, making it a promising approach for training multi-billion-parameter networks efficiently. Our findings shed light on the potential of low-rank training techniques and their implications for scaling laws. Code is available on GitHub.[2]

1 Introduction

Over the past decade, the machine learning field has been dominated by the trend of training increasingly overparametrized networks or adopting the "stack more layers" approach [32, 21, 27]. The definition of a large network has evolved from models with 100 million [46, 39] to hundreds of billions [8, 12] of parameters, which has made computational costs associated with training of such networks prohibitive to most of the research groups. Despite this, the necessity to train models which can have orders of magnitude more parameters than the training examples [8, 12, 16], is poorly understood theoretically [25, 4, 60].

Alternative approaches to scaling, such as more compute-efficient scaling optima [22], retrieval-augmented models [28, 7], and the simple approach of training smaller models for longer [50], have offered new trade-offs. However, they do not bring us closer to understanding why we need overparametrized models and rarely democratize the training of these models. For example, training RETRO [7] requires a complex training setup and infrastructure capable of quickly searching over trillions of tokens, while training LLaMA-6B [50] still requires hundreds of GPUs.

In contrast, approaches like zero-redundancy optimizers [43], 16-bit training [37], 8-bit inference [14], and parameter-efficient fine-tuning (PEFT) [33] have played a crucial role in making large models more accessible. Specifically, PEFT methods have enabled fine-tuning of billion-scale language or diffusion models on consumer hardware. This raises the question: Can these approaches also benefit pre-training?

On the one hand, pre-training is exactly the step that allows for small modifications to the network to adapt it to new tasks. Aghajanyan et al. [1] demonstrated that the rank of the changes required to learn a task decreases the more you pre-train the network. On the other hand, multiple studies

[*]Correspondance to vlad.lialin@gmail.com
[2]github.com/guitaricet/peft_pretraining

Preprint. Under review.

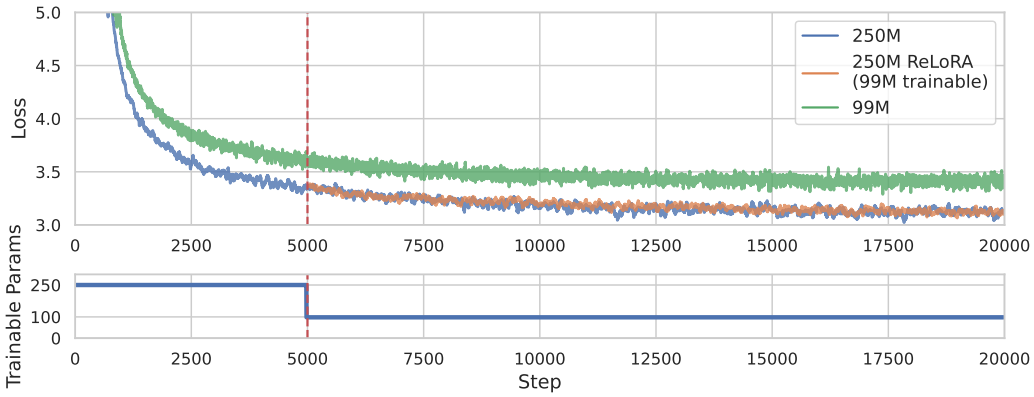

Figure 1: ReLoRA learns a high-rank network through a sequence of low-rank updates. It outperforms networks with the same trainable parameter count and achieves similar performance to training a full network at 100M+ scale. The efficiency of ReLoRA increases with the model size, making it a viable candidate for multi-billion-parameter training.

have demonstrated the simplicity of features extracted and utilized by language and vision models, along with their low intrinsic dimensionality [31, 17, 42, 47]. For instance, attention patterns in transformers [51] often exhibit a small rank, which has been successfully leveraged to develop more efficient variants of attention [52, 11]. Moreover, overparametrization is also not necessary for training. The Lottery Ticket Hypothesis [17] empirically demonstrates that during initialization (or early in training [18]), there exist sub-networks – winning tickets – that when trained in isolation reach the performance of the full network.

In this study, we focus on low-rank training techniques and introduce ReLoRA that uses low-rank updates to train a high-rank network. We empirically demonstrate that ReLoRA performs a high-rank update and achieves performance similar to regular neural network training. The components of ReLoRA include initial full-rank training of the neural network (similar to Frankle et al. [18]), LoRA training, restarts, a jagged learning rate schedule, and partial optimizer resets. We evaluate ReLoRA on transformer language models up to 350M parameters. We chose to focus on autoregressive language modeling, as this approach has demonstrated its universality in most of the applications of neural networks [41, 56, 3, 35, 10]. Finally, we observe that the efficiency of ReLoRA increases with model size, making it a viable option for efficient training of multi-billion-parameter networks.

Each experiment in this study has used no more than 8 GPU days of compute.

2 Related work

Scaling versus Efficiency The relationship between overparametrization and neural network trainability and generalization has been extensively studied [59, 5, 17, 38, 47], yet it remains a mystery [60]. Moreover, scaling laws [27, 19, 22, 30, 2] demonstrate a simple and strong power-law dependence between network size and its performance across a variety of modalities. This finding not only supports overparametrization but also encourages the training of extraordinarily resource-intensive neural networks [8, 12, 16]. Nonetheless, the Lottery Ticket Hypothesis [17, 18] suggests that overparametrization could, in principle, be minimized. Specifically, it shows that *early in training, subnetworks exist that can be trained to achieve the performance of the full network (winning tickets).*

Parameter-efficient fine-tuning Aghajanyan et al. [1] found that pre-training reduces the amount of change to the network, or its intrinsic dimensionality, to learn a new task through fine-tuning. I.e., larger networks or networks pre-trained on more data require smaller modifications in terms of the rank of the range to learn a new task. This explains the success of parameter-efficient fine-tuning methods [33] and has also motivated the development of low-rank fine-tuning methods such as LoRA [23] and Compacter [36].

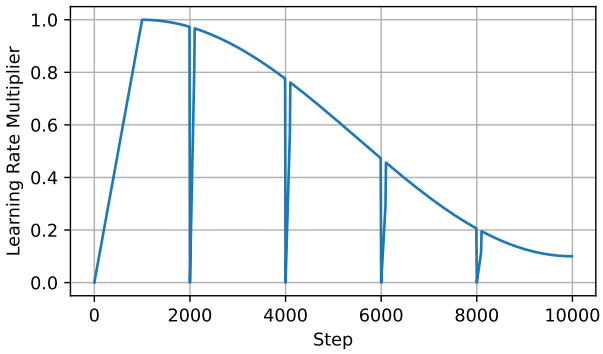

Figure 2: Jagged cosine scheduler used in ReLoRA. On every ReLoRA reset, we set the learning rate to zero and perform a quick (50-100 steps) learning rate warmup back to the cosine schedule.

Low-rank neural network training Training low-rank representations has been explored in the context of CNN compression, regularization, and efficient training [24, 26, 49, 44, 34, 57]. However, most of these methods are either specific to CNNs, do not scale well, or have not been evaluated on large transformers [51] with hundreds of millions of parameters, which can benefit greatly from efficient training. While transformers have been shown to have a low-rank internal dimensionality and representations [1, 52], the study by Bhojanapalli et al. [6] demonstrated that the low rank of key and query projections in multi-head attention bottlenecks the performance of transformers. Our own experiments (Section 3) also demonstrate that low-rank transformers perform significantly worse compared to the full-rank baseline and ReLoRA.

3 Method

Let's start by revisiting linear algebra-101. In particular, we are interested in the rank of the sum of two matrices:

$$\text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B). \tag{1}$$

This bound on the rank of the sum is tight: for a matrix $\mathbf{A}, \text{rank}(\mathbf{A}) < dim(\mathbf{A})$, there exists $\mathbf{B}$, $\text{rank}(\mathbf{B}) < dim(\mathbf{B})$ such that sum of the matrices has a higher rank than either $\mathbf{A}$ or $\mathbf{B}$. We want to exploit this property to make a flexible parameter-efficient training method. We start with LoRA [23] which is a parameter-efficient fine-tuning method based on the idea of low-rank updates. LoRA can be applied to any linear operation parametrized through $W \in \mathbb{R}^{m \times n}$. Specifically, LoRA decomposes the weight update δW into a low-rank product $W_A W_B$ as shown in Equation 2, where $s \in \mathbb{R}$ is a fixed scaling factor usually equal to $\frac{1}{r}$.

$$\begin{aligned} \delta W &= s W_A W_B \\ W_A &\in \mathbb{R}^{\text{in} \times r}, W_B \in \mathbb{R}^{r \times \text{out}} \end{aligned} \tag{2}$$

In practice, LoRA is usually implemented by adding new trainable parameters W_A and W_B, which could be merged back into the original parameters after training. Thus, even though Equation 1 allows the total update over training time $\sum_t \delta W_t$ to have a higher rank than any of the individual matrices, LoRA implementations are restricted by the rank $r = \max_{W_A, W_B} \text{rank}(W_A W_B)$.

If we could restart LoRA, meaning we merge W_A and W_B during training and reset the values of these matrices, we could increase the total rank of the update. Doing this multiple times brings the total neural network update to

$$\Delta W = \sum_{t=0}^{T_1} \delta W_t + \sum_{t=T_1}^{T_2} \delta W_t + \cdots + \sum_{t=T_{N-1}}^{T_N} \delta W_t = s W_A^1 W_B^1 + s W_A^2 W_B^2 + \cdots + s W_A^N W_B^N \tag{3}$$

where the sums are independent enough, meaning that $\text{rank}(W_A^i W_B^i) + \text{rank}(W_A^j W_B^j) \geq r$.

However, implementing restarts is not trivial in practice and requires certain modifications to the optimization procedure. Naïve implementation causes the model to diverge right after the restart. Unlike plain stochastic gradient descent, which solely relies on the value of the gradient at the current optimization timestep, Adam [29] update is guided mainly by the first and second moments of the gradient accumulated over the previous steps. In practice, gradient moment smoothing parameters β_1 and β_2 are usually very high $0.9 - 0.999$. Let's assume that at the reinitialization boundary W_A^1 and the corresponding gradient moments m_A and v_A, are full-rank (r). Then, after the merge-and-reinit, continuing to use old gradient moments for W_A^2 will guide it in the same direction as W_A^1 and optimize the same subspace.

To resolve this issue, we propose ReLoRA. ReLoRA performs a partial reset of the optimizer state during merge-and-reinit and sets the learning rate to 0 with a subsequent warmup. Specifically, we set 99% of low-magnitude optimizer state values to zero and use a jagged-cosine learning rate schedule (Figure 2). Our ablation studies (Table 3) show that both of these modifications are required to improve the performance over vanilla LoRA.

To reiterate, ReLoRA is a low-rank training method inspired by LoRA that uses restarts to increase the effective rank of the update, uses partial optimizer reset, and a jagged scheduler to stabilize training and warm starts. All of this allows ReLoRA to achieve performance comparable to full-rank training, especially in large transformer networks, by only training a small set of parameters at a time. ReLoRA is described in Algorithm 1.

Enhancing computational efficiency Unlike other low-rank training techniques [44, 49], ReLoRA follows the LoRA approach by maintaining the frozen weights of the original network and adding new trainable parameters. At first glance, this may appear computationally inefficient; however, the differentiation between frozen and trainable parameters plays a crucial role in parameter-efficient fine-tuning [33].

These methods achieve significant improvements in training time and memory efficiency by reducing the size of the gradients and the optimizer states. Notably, Adam states consume twice as much memory as the model weights. Moreover, it is common practice to maintain gradient accumulation buffers in 32-bit precision for large networks, thereby adding significant overhead to the memory consumption of gradients.

By substantially reducing the number of trainable parameters, ReLoRA enables the utilization of larger batch sizes, maximizing hardware efficiency. Additionally, it reduces the bandwidth requirements in distributed setups, which are often the limiting factor in large-scale training.

Furthermore, since the frozen parameters are not being updated between restarts, they can be kept in a low-precision quantized format, further reducing their memory and computational impact. This additional optimization contributes to overall improved efficiency in terms of memory utilization and computational resources of ReLoRA and increases at scale.

4 Experiments

To evaluate the effectiveness of ReLoRA, we apply it to train a transformer language model on the C4 dataset [41] using various model sizes: 60M, 130M, 250M, and 350M. Language modeling has been shown to be a fundamental task in machine learning [40], it enables text and image classification [56], translation [8], programming [9], in-context learning, step-by-step reasoning [54], and many other emergent abilities [53]. Given its significance, we focus solely on language modeling for the purposes of this paper.

Architecture and training hyperparameters Our architecture is based on transformer [51] and closely resembles LLaMA [50]. Namely, we use pre-normalization, RMSNorm [58], SwiGLU activations [45], $\frac{8}{3}h$ fully-connected hidden state size [50], and rotary embeddings [48]. For all LoRA and ReLoRA experiments, we use rank $r = 128$ as our initial experiments showed it to have the best perplexity/memory tradeoff. All hyperparameters are presented in Table 1.

We use bfloat16 for all floating point operations and Flash attention [13] for effective attention computation. Compared to attention in LLaMA, which uses float32 for softmax computation, this increased training throughput by 50-100% without any training stability issues.

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 scheduled according to a jagged scheduler, and finally, 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_A^i, W_B^i)
    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**
        $W^i \leftarrow (W^i + sW_A^i W_B^i)$
        $W_A^i \leftarrow$ kaiming_init(W_A^i); $W_B^i \leftarrow 0$
        $M_{W_A^i} \leftarrow$ prune$(M_{W_A^i})$; $V_{W_A^i} \leftarrow$ prune$(V_{W_A^i})$
      end if
    end for
    Start η warmup
  end if
end for
return θ

Most of our models were trained on 8 RTX 4090 for one day or less. Due to computational constraints, we train much smaller models than LLaMA, with the largest model having 350M parameters, the same as BERT Large [15]. We select the number of pre-training tokens based on the Chinchilla scaling laws [22] for all models, except for the largest one, which we train for 6.8B tokens while 9.5B tokens are Chinchilla-optimal.

ReLoRA and baselines setup In our low-rank training experiments, ReLoRA replaces all attention and fully-connected network parameters, while keeping the embeddings full-rank. The RMSNorm parametrization remains unchanged. Since ReLoRA-wrapped models have fewer trainable parameters than full-rank training, we include a Control baseline, which is a full-rank transformer with the same number of trainable parameters as ReLoRA.

We initialize ReLoRA from a checkpoint of full-rank training at 5,000 update steps and reset it every 5,000 steps thereafter, 3 times in total. After each reset, 99% of the optimizer state is pruned based on magnitude, and the loss is warmed up for the next 100 iterations. ReLoRA parameters are reinitialized following LoRA best practices, Kaiming initialization [20] for A-matrix, and zeros for B-matrix. In case of not using the restarts, the B-matrix also uses Kaiming initialization to avoid gradient-symmetry issues.

Params	Hidden	Heads	Layers	Learning rate	Batch (tokens)	Seq. len.	Tokens
60M	512	8	8	1e-3	122K	256	1.2B
130M	768	12	12	1e-3	154K	256	2.6B
250M	768	16	24	5e-4	590K	512	6.8B
350M	1024	16	24	5e-4	590K	512	6.8B

Table 1: Hyperparameters of the language models trained in this study.

	60M	130M	250M	350M
Full training	33.81 (60M)	23.65 (130M)	22.39 (250M)	18.66 (350M)
Control	36.52 (43M)	27.30 (72M)	25.43 (99M)	23.65 (130M)
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)
ReLoRA	**34.46** (43M)	**25.04** (72M)	**22.48** (99M)	**19.32** (125M)
Training tokens (billions)	1.2	2.6	6.8	6.8

Table 2: Comparing perplexities between baseline methods and ReLoRA (lower is better). 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. Low-rank training is **bold** if it outperforms the Control baseline.

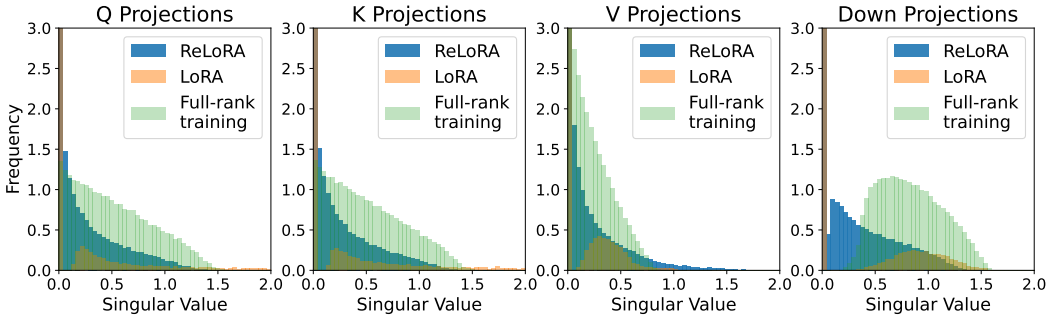

Figure 3: Singular values spectra of the weight difference between ReLoRA and LoRA at 5,000 iterations (warm start) and 20,000 iterations. ReLoRA exhibits a closer resemblance to full-rank training singular values than LoRA, indicating its effectiveness in approximating full-rank behavior.

5 Results

Parameter-efficient pre-training Our main results are resented in Table 2. ReLoRA significantly outperforms low-rank LoRA training demonstrating the effectiveness of our proposed modifications (ablated in Section 3). Furthermore, ReLoRA achieves similar performance to full-rank training, and the performance gap diminishes as network size increases.

High-rank training through low-rank updates To determine whether ReLoRA performs a higher rank update than LoRA we plot the singular value spectrum of the difference between warm-start weights and the final weights for ReLoRA, LoRA, and full-rank training. Figure 3 illustrates significant qualitative differences between LoRA and ReLoRA for the singular values of W_Q, W_K, W_V, and W_{down}.

While most of the singular values for LoRA are zero (Figure 4) with a noticeable number of exceptionally high values above 1.5, ReLoRA exhibits a higher distribution mass between 0.1 and 1.0, reminiscent of full-rank training. This observation emphasizes the significance of high-rank updates and demonstrates the qualitative efficacy of ReLoRA, which accomplishes a high-rank update by performing multiple low-rank updates.

5.1 Ablation studies

We conduct ablation studies on all four crucial components of ReLoRA: restarts, jagged schedule, optimizer resets, and warm starts, utilizing the 130M-sized model. The results are presented in Table 3. In this section, we will focus on and analyze certain combinations of these components.

LoRA ReLoRA, without the aforementioned components, is essentially equivalent to training a low-rank network parameterized by LoRA. This approach yields remarkably high perplexity,

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

Table 3: Ablation studies of ReLoRA (130M models). Restarts and warm starts are essential for good performance. Using restarts and optimizer reset without a jagged schedule causes the model to diverge.

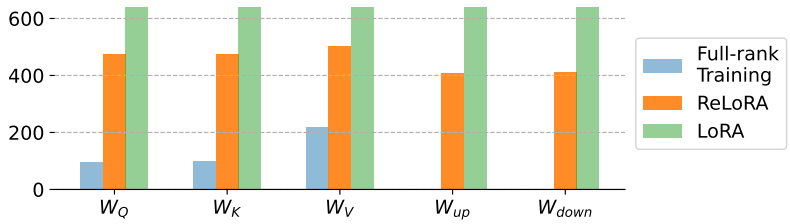

Figure 4: Number of singular values < 0.1 in attention and FCN projection matrices.

indicating that a simple matrix decomposition has significantly different training dynamics from full-rank training.

Adding restarts and optimizer resets ReLoRA, without a jagged schedule and optimizer reset, performs similarly to LoRA because old optimizer states force the newly initialized parameters into the same subspace as the prior weights, limiting the model's capacity. However, doing a naive optimizer reset with ReLoRA causes the model to diverge. A jagged schedule helps to stabilize training and has a positive impact on the mixture. In our initial experiments, we also observed that a combination of partial optimizer reset and jagged scheduler allows for a quicker warm-up, as low as 50 steps, instead of hundreds of steps required when the optimizer is initialized from scratch.

Warm start The warm start shows the most significant improvement, dropping perplexity by almost 10 points. To investigate whether post-warmup training contributes to the loss, we measured the perplexity of the warmed-up network, which equals 27.03. It outperforms all low-rank methods except for our final ReLoRA recipe but still demonstrates a significant difference from the final network. This demonstrates the importance of early training, similar to the concept of the lottery ticket hypothesis with rewinding [18].

6 Conclusion

In this paper, we investigated low-rank training techniques for large transformer language models. We first examined the limitations of a simple low-rank matrix factorization (LoRA) approach and observed that it struggles to effectively train high-performing transformer models. To address this issue, we proposed a novel method called ReLoRA, which leverages the rank of sum property to train a high-rank network through multiple low-rank updates. Similar to the lottery ticket hypothesis with rewinding, ReLoRA employs a full-rank training warm start before transitioning to ReLoRA. Additionally, ReLoRA introduces a merge-and-reinit (restart) strategy, a jagged learning rate scheduler, and partial optimizer resets, which collectively enhance the efficiency of ReLoRA and bring it closer to full-rank training, **particularly in large networks**. ReLoRA efficiency increases with the network size making it a viable candidate for multi-billion-scale training.

We firmly believe that the development of low-rank training methods holds great promise for improving the efficiency of training large language models and neural networks in general. Furthermore, low-rank training has the potential to provide valuable insights for the advancement of deep learning theories, aiding our understanding of neural network trainability through gradient descent and their exceptional generalization capabilities in the overparametrized regime.

7 Limitations and Future Work

Scaling beyond 350M Due to limited computational resources, our experiments were constrained to training language models with up to 350M parameters. Nonetheless, ReLoRA already demonstrates promising results at this scale. However, we anticipate its true potential will be realized in the 1B+ parameter region. Additionally, while the 350M model outperforms the Control baseline, it does not continue the trend of narrowing the gap between ReLoRA and full-rank training. We attribute this to suboptimal hyperparameter choice, which requires further investigation.

Furthermore, in 60-350M experiments, even though ReLoRA significantly reduces the number of trainable parameters, we did not observe substantial improvements in memory and computation for the networks of this size. To evaluate the efficiency of our current implementation at a larger scale, we trained the 1.3B-parameter model for a small number of iterations to estimate memory and compute improvements of ReLoRA. At this scale, we observe *30% memory consumption reduction and 52% training throughput increase*. We expect to observe even bigger improvements over the full-training baseline for larger networks since the number of trainable parameters for ReLoRA, similar to LoRA, increases at a much slower rate compared to the number of frozen parameters. ReLoRA implementation could be further improved by effectively utilizing gradient checkpointing for ReLoRA layers, custom backward functions, and converting frozen model weights to int8 or int4 quantized format [14].

Comparison to other low-rank training methods A number of approaches to low-rank training have been explored with other model architectures in earlier work [44, 49, 55]. Two aspects set our work apart from these earlier efforts. First, the approach we propose performs high-rank updates through low-rank training. Second, our work demonstrates competitiveness of the low-rank training methods in large-scale transformer language models with 100M+ parameters.

Acknowledgments and Disclosure of Funding

We would like to say thanks to Stability.ai, Eleuther.ai, Google Cloud for Research Program, Eric Lehman, and Artem Krovosheev for helping with computational resources for this paper. Special thanks to Jason Phang, Hailey Schoelkopf, and Enrico Shippole for their technical advice.

This work was funded in part by an Amazon Alexa AI research award to Anna Rumshisky.

References

[1] A. Aghajanyan, S. Gupta, and L. Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.568. URL `https://aclanthology.org/2021.acl-long.568`.

[2] A. Aghajanyan, L. Yu, A. Conneau, W.-N. Hsu, K. Hambardzumyan, S. Zhang, S. Roller, N. Goyal, O. Levy, and L. Zettlemoyer. Scaling laws for generative mixed-modal language models, 2023.

[3] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

[4] Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 242–252. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/allen-zhu19a.html`.

[5] M. Belkin, D. J. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116:15849 – 15854, 2018.

[6] S. Bhojanapalli, C. Yun, A. S. Rawat, S. Reddi, and S. Kumar. Low-rank bottleneck in multi-head attention models. In *International Conference on Machine Learning*, pages 864–873. PMLR, 2020.

[7] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. De Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. Rae, E. Elsen, and L. Sifre. Improving language models by retrieving from trillions of tokens. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/borgeaud22a.html`.

[8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[9] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code. 2021.

[10] J. Cho, J. Lei, H. Tan, and M. Bansal. Unifying vision-and-language tasks via text generation. In *International Conference on Machine Learning*, pages 1931–1942. PMLR, 2021.

[11] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlós, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL `https://openreview.net/forum?id=Ua6zuk0WRH`.

[12] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. M. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. C. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. García, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Díaz, O. Firat, M. Catasta, J. Wei, K. S. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311, 2022.

[13] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Re. Flashattention: Fast and memory-efficient exact attention with IO-awareness. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=H4DqfPSibmx`.

[14] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=dXiGWqBoxaD`.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

[16] W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23(1), jan 2022. ISSN 1532-4435.

[17] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

[18] J. Frankle, G. Karolina Dziugaite, D. M. Roy, and M. Carbin. Stabilizing the lottery ticket hypothesis. *arXiv e-prints*, pages arXiv–1903, 2019.

[19] B. Ghorbani, O. Firat, M. Freitag, A. Bapna, M. Krikun, X. García, C. Chelba, and C. Cherry. Scaling laws for neural machine translation. *ArXiv*, abs/2109.07740, 2021.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. URL http://arxiv.org/abs/1502.01852.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[22] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, O. Vinyals, J. W. Rae, and L. Sifre. An empirical analysis of compute-optimal large language model training. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=iBBcRUlOAPR.

[23] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[24] Y. Idelbayev and M. A. Carreira-Perpinan. Low-rank compression of neural nets: Learning the rank of each layer. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8046–8056, 2020. doi: 10.1109/CVPR42600.2020.00807.

[25] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8580–8589, Red Hook, NY, USA, 2018. Curran Associates Inc.

[26] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014. doi: http://dx.doi.org/10.5244/C.28.88.

[27] J. Kaplan, S. McCandlish, T. J. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *ArXiv*, abs/2001.08361, 2020.

[28] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HklBjCEKvH.

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[30] T. Klug and R. Heckel. Scaling laws for deep learning based image reconstruction. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=op-ceGueqc4.

[31] O. Kovaleva, A. Romanov, A. Rogers, and A. Rumshisky. Revealing the dark secrets of bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, 2019.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[33] V. Lialin, V. Deshpande, and A. Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning, 2023.

[34] R. Lin, C.-Y. Ko, Z. He, C. Chen, Y. Cheng, H. Yu, G. Chesi, and N. Wong. Hotcake: Higher order tucker articulated kernels for deeper cnn compression. In *2020 IEEE 15th International Conference on Solid-State & Integrated Circuit Technology (ICSICT)*, pages 1–4, 2020. doi: 10.1109/ICSICT49897.2020.9278257.

[35] J. Lu, C. Clark, R. Zellers, R. Mottaghi, and A. Kembhavi. UNIFIED-IO: A unified model for vision, language, and multi-modal tasks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=E01k9048soZ.

[36] R. K. mahabadi, J. Henderson, and S. Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=bqGK5PyI6-N.

[37] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1gs9JgRZ.

[38] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021, 2019.

[39] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[40] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[41] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[42] A. Raganato, Y. Scherrer, and J. Tiedemann. Fixed encoder self-attention patterns in transformer-based machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 556–568, 2020.

[43] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16, 2020. doi: 10.1109/SC41405.2020.00024.

[44] S. Schotthöfer, E. Zangrando, J. Kusch, G. Ceruti, and F. Tudisco. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20051–20063. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/7e98b00eeafcdaeb0c5661fb9355be3a-Paper-Conference.pdf.

[45] N. Shazeer. Glu variants improve transformer, 2020.

[46] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.1556.

[47] S. P. Singh, G. Bachmann, and T. Hofmann. Analytic insights into structure and rank of neural network hessian maps. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=otDgw7LM7Nn.

[48] J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *ArXiv*, abs/2104.09864, 2021.

[49] Y. Sui, M. Yin, W. Yang, Y. Gong, J. Xiao, H. Phan, D. Ding, X. Xu, S. Liu, Z. Chen, and B. Yuan. ELRT: Towards efficient low-rank training for compact neural networks, 2023. URL https://openreview.net/forum?id=TC39w69m8bB.

[50] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[52] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[53] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=yzkSU5zdwD. Survey Certification.

[54] J. Wei, X. Wang, D. Schuurmans, M. Bosma, brian ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

[55] H. Yang, M. Tang, W. Wen, F. Yan, D. Hu, A. Li, H. H. Li, and Y. Chen. Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2899–2908, 2020.

[56] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu. Coca: Contrastive captioners are image-text foundation models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=Ee277P3AYC.

[57] X. Yuan, P. H. P. Savarese, and M. Maire. Growing efficient deep networks by structured continuous sparsification. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=wb3wxCObbRT.

[58] B. Zhang and R. Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

[59] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sy8gdB9xx.

[60] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64:107 – 115, 2021.

A Downstream evaluation

We additionally perform a downstream evaluation of 350M models pretraiend either using ReLoRA or regular full-rank training. For a baseline, we additionally compare them to a randomly-initialized model.

	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Full-rank pretrained	35.19	85.01	80.38	53.79	63.59
Not pretrained	5.39	26.94	73.13	53.07	39.63
ReLoRA	36.12	84.75	81.41	52.35	63.66

Table 4: Downstream evaluation results on GLUE benchmark (dev set).

B Smaller warm start period

Table 2 demonstrates that ReLoRA consistently outperforms the warmed-started LoRA baseline. To provide a more contrasting example, we performed additional pre-training experiments starting from just 2K warm-started network. Figure 5 shows a significant performance gain with ReLoRA over LoRA by 1.4 ppl points (ppl 23.64 vs 25.08). While the absolute performance of ReLoRA is lower compared to full-rank training in this context, these experiments validate our initial hypothesis that LoRA restarts positively impact performance.

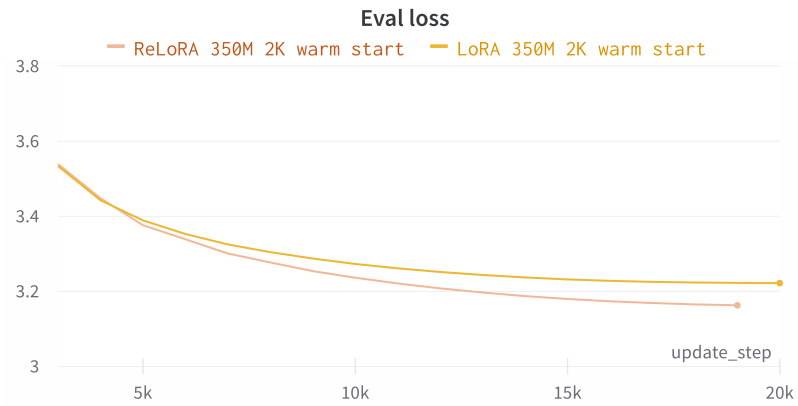

Figure 5: ReLoRA significantly outperforms LoRA when started from an early (2K steps) checkpoint.

	60M	130M	250M	350M
Full training	33.81 (60M)	23.65 (130M)	22.39 (250M)	18.66 (350M)
Control	36.52 (43M)	27.30 (72M)	25.43 (99M)	23.65 (130M)
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)
ReLoRA	**34.46** (43M)	**25.04** (72M)	**22.48** (99M)	**19.32** (125M)
Training tokens (billions)	1.2	2.6	6.8	6.8

Table 2: Comparing perplexities between baseline methods and ReLoRA (lower is better). 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. Low-rank training is **bold** if it outperforms the Control baseline.

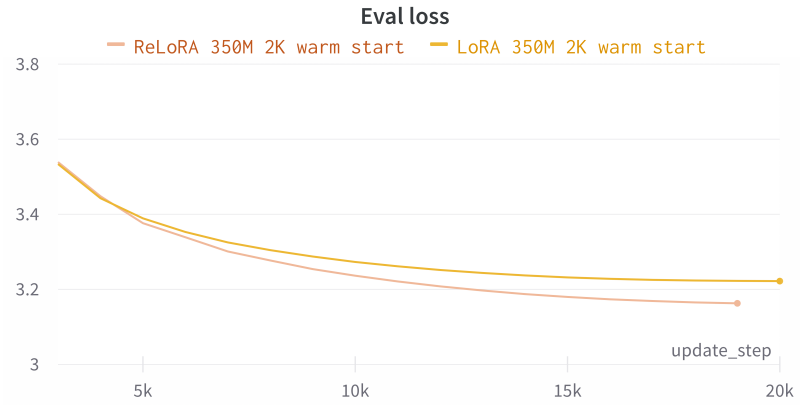

Figure 5: ReLoRA significantly outperforms LoRA when started from an early (2K steps) checkpoint.

	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Full-rank pretrained	35.19	85.01	80.38	53.79	63.59
Not pretrained	5.39	26.94	73.13	53.07	39.63
ReLoRA	36.12	84.75	81.41	52.35	63.66

Table 4: Downstream evaluation results on GLUE benchmark (dev set).