Investigating ReLoRA: Effects on the Learning Dynamics of Small Language Models

Yuval Weiss[*] **David Demitri Africa**
Paula Buttery **Richard Diehl Martinez**
University of Cambridge

Abstract

Parameter-efficient methods like LoRA have revolutionised large language model (LLM) fine-tuning. ReLoRA extends this idea to pretraining by repeatedly merging and reinitialising low-rank adapters, increasing cumulative rank while keeping updates cheap. This aligns well with observations that high-capacity models learn through locally low-rank trajectories that expand over time. By contrast, recent work suggests that small language models (SLMs) exhibit rank deficiencies and under-utilise their available dimensionality. This raises a natural question: can ReLoRA's rank-expanding update rule *steer* SLMs toward healthier learning dynamics, mitigating rank bottlenecks in a capacity-constrained regime? We argue SLMs are an ideal testbed: they train quickly, enable controlled ablations, and make rank phenomena more measurable. We present the first systematic study of ReLoRA in SLMs (11M-66M parameters), evaluating both performance and learning dynamics. Across loss, Paloma perplexity, and BLiMP, we find that ReLoRA underperforms full-rank training, with gaps widening at larger scales. Analysis of proportional effective rank and condition numbers shows that ReLoRA amplifies existing rank deficiencies and induces ill-conditioned updates early in training. Our results suggest that while ReLoRA's merge-and-restart strategy can expand ranks in larger models, it does not straightforwardly translate to capacity-limited SLMs, motivating adaptive-rank or hybrid-rank approaches for low-compute pretraining.

🤗 yuvalw/pico-relora

Yu-val-weiss/pico-relora

1 Introduction

Contemporary work on language modelling has continually prioritised ever-larger scales, delivering remarkable capability gains (Chowdhery et al., 2023; Grattafiori et al., 2024; OpenAI, 2024). However, the computational and environmental costs of large-scale language modelling are substantial (Chowdhery et al., 2023; Grattafiori et al., 2024; Morrison et al., 2024; OpenAI, 2024). Small language models (SLMs) offer a complementary path: they are cheap to train and deploy, easier to study, and attractive for settings where safety, privacy, or energy constraints dominate (Schwartz et al., 2020). However, SLMs lag in quality due to tight capacity limits and brittle optimisation (Mielke et al., 2019; Ettinger, 2020; Kaplan et al., 2020).

A rank-centric view of learning. A growing body of work frames transformer learning through the lens of the rank of model weights. In high-capacity models, training often proceeds via low-rank updates that *gradually* expand effective rank (Aghajanyan et al., 2021; Boix-Adsera et al., 2023). Conversely, SLMs show rank deficiencies and anisotropic representations that restrict gradient flow and under-utilise model dimensionality (Noci et al., 2022; Diehl Martinez et al., 2024; Godey et al., 2024a). From this perspective, methods that shape the rank profile of updates could directly influence optimisation quality and final capability.

LoRA. LoRA (Hu et al., 2022) has been widely used due to its effectiveness and applicability to any model with matrix computations, enabling straightforward fine-tuning of even the largest models (Blattmann et al., 2023; Dettmers et al., 2023; Fomenko et al., 2024). LoRA introduces low-rank adaptation, where the pretrained model weights are frozen, and fine-tuning is performed with trainable rank decomposition matrices at each layer of the architecture, thus significantly reducing the number of trainable parameters.

ReLoRA. ReLoRA, proposed by (Lialin et al., 2023), aims to leverage LoRA's overwhelming success in low-rank adaptation for fine-tuning to

[*]Corresponding author: yw580@cantab.ac.uk

Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP, pages 163–175
November 9, 2025 ©2025 Association for Computational Linguistics

advance efficient pretraining. It does this by injecting low-rank LoRA-style matrices into a language model, then repeatedly merging and reinitialising them throughout the training process, thereby parametrising gradient updates through the merge operations at each restart. The method is tested on models with 60M to 1.3B parameters, improving training speeds and reducing GPU memory footprints, especially at the higher end. However, research for models even smaller than this is limited.

Why ReLoRA might help SLMs. Small transformer models suffer from rank deficiencies in their weight matrices, limiting the subspace they can explore during training (Diehl Martinez et al., 2024). ReLoRA (Lialin et al., 2023) periodically merges low-rank adapters into the base weights and reinitialises them. Summing distinct low-rank updates increases the rank of the *cumulative* update, potentially widening the subspace explored over training. If SLMs struggle because their updates remain trapped in low-rank bottlenecks, then an explicit rank-expansion mechanism could *steer* them toward healthier dynamics, improving sample efficiency and language competence in the low compute regime. This provides the intuition behind our methodology.

Why test this in SLMs? SLMs are fast to train, enabling careful ablations across model structure alongside detailed diagnostics such as effective rank and condition number that would otherwise be prohibitively costly in billion-parameter models. Their capacity limits also amplify rank phenomena, making both benefits and failure modes easier to detect. Therefore, SLMs are a sensitive and economical *sandbox* for understanding how rank-structured updates shape learning.

This work: *research question*. In this work, we ask whether ReLoRA's rank-expanding merge-and-restart methodology actually helps in the capacity-constrained regime. In other words, does ReLoRA **boost** performance or **drag** it down? The case for either can be summarised as follows:

👍 **Boost**: By strategically resetting and merging LoRA matrices, ReLoRA aggregates multiple low-rank steps, which may widen those bottlenecks and enable smaller models to capture more complex patterns

👎 **Drag**: However, ReLoRA may also have the opposite effect: further reducing the (already limited) representational capacity of SLMs.

This work: *outline of methodology and results*. We extend a Llama-style SLM (Touvron et al., 2023; Diehl Martinez, 2025) with ReLoRA and run matched ablations at 11M and 66M parameters. We evaluate training loss, Paloma perplexity (Magnusson et al., 2024), and BLiMP (Warstadt et al., 2020), and additionally probe learning dynamics via proportional effective rank (PER) and condition numbers of both weights and updates. Contrary to the motivating intuition, ReLoRA *does not* boost SLMs: it underperforms full-rank training and reinforces rank deficiencies, with early training marked by highly ill-conditioned updates. We thus conclude that the mechanism that helps larger models does not straightforwardly transfer to SLMs.

This work: *contributions*. Our contributions are as follows:

(1) Novel systematic evaluation of ReLoRA for small language models (SLMs). We investigate ReLoRA on 11M and 66M parameter models, filling a critical gap in understanding its behaviour for low-compute domains. We make two public code contributions: a public HuggingFace spaceto be used for evaluating LMs on the BLiMP task through the `evaluate` library and a public fork, `pico-relora`, of `pico-train` (Diehl Martinez, 2025) extending `pico-decoder` with ReLoRA.

(2) Comprehensive analysis of learning dynamics. We compute proportional effective rank (PER) and condition numbers of weights and gradient updates throughout training, revealing that ReLoRA consistently reduces rank and induces highly ill-conditioned updates in SLMs.

(3) 👎 Drag: Evidence of performance degradation and training instability. Across loss, Paloma perplexity, and BLiMP evaluation, ReLoRA underperforms its full-rank baseline, with widening gaps as model size increases from the `tiny` to `small` scales. Our findings suggest that low-rank strategies do not trivially transfer to SLMs and thus motivate hybrid or adaptive-rank approaches for future low-resource model training.

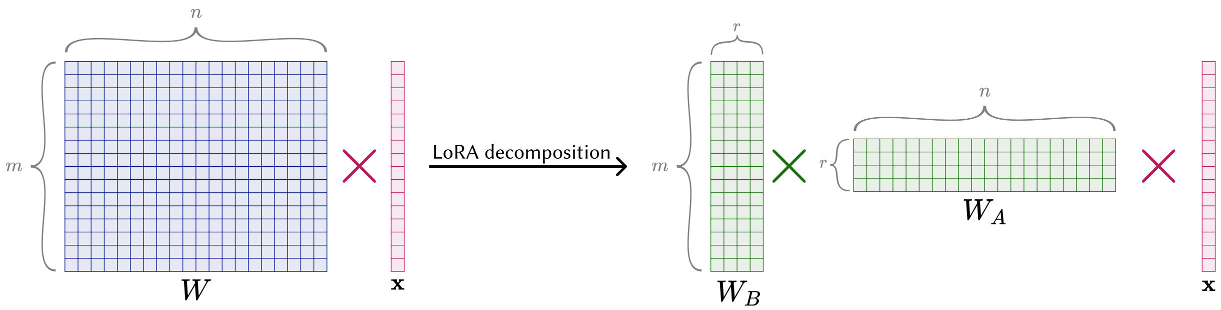

Figure 1: LoRA decomposition (Hu et al., 2022), which can be applied to any linear operation parameterised by a matrix

2 Background: ReLoRA

We now introduce ReLoRA, a method utilising low-rank updates to train high-rank neural networks more efficiently (Lialin et al., 2023).

Motivation: LoRA to ReLoRA. ReLoRA is based on LoRA, a vastly successful parameter-efficient fine-tuning technique which leverages 'LoRA decomposition' matrices to effectively fine-tune even the very largest models (Hu et al., 2022; Fomenko et al., 2024). Given LoRA's success, (Lialin et al., 2023) propose ReLoRA, which applies the same decomposition methodology to the *pretraining* of an LM, providing competitive performance for significantly less compute.

LoRA decomposition. In a neural network, a linear operation parameterised by $W \in \mathbb{R}^{m\times n}$ applied to a token $\mathbf{x} \in \mathbb{R}^n$ has a gradient denoted ΔW. After a gradient update, the output of the linear operation is changed by $(W + \Delta W)x = Wx + \Delta W x$. LoRA proposes decomposing the gradient update ΔW to two matrices $W_A \in \mathbb{R}^{r\times n}$ and $W_B \in \mathbb{R}^{m\times r}$, for some very small LoRA parameter $r \ll \min(m, n)$ and scaling parameter s, as shown in Equation 1 and Figure 1. These matrices are known as adapters.

$$\Delta W = sW_B W_A \tag{1}$$

For small r, the number of parameters being directly updated is $r(m + n) < m \cdot n$.

LoRA is implemented by injecting trainable linear layers parameterised by W_A and W_B into the model. Throughout this work, we refer to W_A and W_B collectively as the LoRA module, and W as the base matrix.

ReLoRA extends LoRA decomposition by introducing restarts. This is motivated by a core property of matrices: for some matrix A, there exists a matrix B such that

$$\text{rank}(A + B) \geq \max(\text{rank}(A), \text{rank}(B)) \tag{2}$$

This means that combining different low-rank updates can result in a higher-rank update overall.

ReLoRA proposes periodically restarting LoRA's low-rank adapters. At each restart, the low-rank matrices W_A and W_B are merged back into W (by adding $W_B W_A$ to W), and reinitialised for further training. After N such restarts, the cumulative update ΔW_{eff} is thus

$$\Delta W_{\text{eff}} = s \sum_{i=1}^{N} W_B^i W_A^i \tag{3}$$

where each pair of W_A^i and W_B^i comes from a distinct stage of ReLoRA in between each restart, and s is a scaling parameter. This assumes that the LoRA modules have identical rank r. The equation only holds for AdamW (Loshchilov and Hutter, 2019) without weight decay on W, thus requiring the optimisation modifications described below.

This strategy effectively increases the rank of the total update ΔW_{eff} even though individual terms remain low-rank. As a result, ReLoRA can represent more complex updates than LoRA with the same number of trainable parameters per step, achieving higher representational effectiveness while maintaining parameter efficiency. The motivation for this approach is grounded in prior observations that neural network training often operates in a locally low-rank regime (Aghajanyan et al., 2021; Boix-Adsera et al., 2023). Understanding how these low-rank dynamics evolve requires situating ReLoRA within the broader literature on rank growth and intrinsic dimensionality.

3 Background: Rank Dynamics

Lialin et al. (2023) measure the singular value spectra produced by models trained with ReLoRA and find that these models resemble models trained with normal full-rank training, particularly when comparing larger singular values. While these results suggest that ReLoRA can retain some of the representational richness of full-rank training, its behaviour in much smaller models remains untested.

This is particularly relevant in light of prior work on intrinsic dimensionality, which measures the minimal number of parameters required to solve a task to a given accuracy. Aghajanyan et al. (2021) show that large pretrained LMs occupy surprisingly low-dimensional subspaces when fine-tuned, sometimes requiring only a few hundred effective parameters. Yet these measurements are limited to a 125M-parameter RoBERTa$_{\text{BASE}}$ (Liu et al., 2019) and focus on fine-tuning rather than pretraining. In the pretraining domain, Diehl Martinez et al. (2024) find that smaller models in general tend to use their available capacity less effectively than larger models.

Relatedly, studies of rank dynamics in transformers have shown two interesting effects: first, weight matrices tend to increase in effective rank gradually over training (Boix-Adsera et al., 2023), expanding their representational capacity; second, at the same time, "rank collapse" at initialisation can severely limit early gradient flow in key and query projections (Noci et al., 2022). These findings suggest that the ability to build rank over time is important for model quality. How ReLoRA intersects with these findings in the context of small language models is an open question and the subject of this paper. We hypothesise two possible effects.

ReLoRA's repeated low-rank resets could either **boost** performance and facilitate rank growth by exploring new subspaces, or **drag** performance by exacerbating rank bottlenecks in small models. Understanding which effect dominates requires probing ReLoRA in precisely these capacity-limited regimes.

4 Methodology

The model implementations and training code are forked from `pico-train` (Diehl Martinez, 2025), a lightweight framework for training language models. `pico-train` implements `pico-decoder`, a Llama-style decoder (Touvron et al., 2023) illustrated and described in Figure 7 in Appendix A. `pico-decoder` is extended with an implementation of ReLoRA (Lialin et al., 2023) to form `pico-relora`. The comparison of these two models forms the basis of the methodology presented in this paper.

Model	Trainable params	Total params
`t-dec`	11,282,784	11,282,784
`s-dec`	64,595,328	64,595,328
`t-rel`	10,060,128	11,682,144
`s-rel`	40,240,512	66,192,768

Table 1: Trainable and total numbers of parameters for the `tiny` and `small` models. `t` = `tiny`, `s` = `small`, `dec` = `decoder`, `rel` = `relora`.

4.1 Experimental setup

The experimental setup is as follows: We execute two `pico-relora` training runs at `tiny` and `small` scales to compare to equivalent (other than the presence of ReLoRA) baseline `pico-decoder` models. This forms a targeted ablation study, consisting of four runs at two scales, by which the behaviour of ReLoRA in small language models can be systematically analysed. The runs' respective numbers of trainable and total parameters are shown in Table 1.

The runs were executed for 20,000 batch steps each, resulting in each model encountering a total of 41.9B tokens throughout training. Training infrastructure and experiment runtimes are described in Appendix B.

Table 2 depicts the configuration parameters for ReLoRA, with the full configurations shown in Tables 4, 5 and 6 of Appendix C. These are identical between the two runs. ReLoRA modules are injected into each linear layer of the attention and feed-forward layers. The modules are set to use a fixed (trainable scaling is set to `False`) scaling parameter $s = \frac{\alpha}{r} = \frac{32}{16} = 2$, a dropout probability of 0.1 and an internal LoRA parameter r of 16. ReLoRA resets are configured to occur every 2000 optimiser steps. These values are based on the ones in Lialin et al. (2023). We omit the 20k step full-rank warm-start for a purer analysis of learning dynamics and to avoid the additional computational overhead.

ReLoRA restarts require modifications to the optimiser and learning rate scheduler. `pico-decoder` uses AdamW (Loshchilov and Hutter, 2019) as its optimiser, which includes moments accumulated over previous gradient update steps in its parameter update calculations. Update step $i+1$

Parameter	Value
Target modules	`attention & swiglu`
Reset frequency	2000
r	16
Trainable scaling	`False`
α	32
Dropout	0.1

Table 2: ReLoRA configuration for training runs. r is the LoRA parameter, and α configures the scaling parameter s according to $s = \frac{\alpha}{r}$

will thus still be guided by gradients calculated in step i. Therefore, ReLoRA randomly prunes a (configurable) proportion of the optimiser states (around 99% of them) to zero to prevent continuing in the same trajectory as before the reset, which could hinder learning new subspaces. The learning rate is simultaneously set to zero to avoid divergent losses and linearly 're-warmed' according to a jagged cosine scheduler. The two learning rate schedulers are illustrated in Figure 8.

4.2 Training and evaluation datasets: Dolma, Paloma and BLiMP

Training on Dolma, a three-trillion-token English dataset. `pico-train` is set up to use a pre-tokenised, pre-shuffled version of Dolma (Soldaini et al., 2024), an open-source, three-trillion-token English dataset.[1] Dolma is composed of scientific writing, website content, computer code, books in the public domain, social media data and material from encyclopedias. Dolma considers both ethical and legal issues throughout the data curation process, avoiding sources that contain copyrighted materials or personally identifiable information.

Perplexity is computed on the Paloma benchmark dataset. In this work, perplexity is computed on Paloma, a benchmark dataset that measures LM fit over 546 different English and computer code domains (Magnusson et al., 2024). The goal of Paloma is to avoid the assumption that an LM suited to one domain will generalise well to others. It uses stratified subsampling based on empirical estimates of variance, ensuring domains are equally represented. Paloma contains 123,683,201 tokens across its test and validation splits; thus, a subsampled version of the corpus, termed `palomy-tinsy`[2] is used. Like `pretokenized-dolma`, `palomy-tinsy` is pre-tokenised and pre-shuffled. It consists of 1.44 thousand data points in a single validation split.

Evaluating linguistic understanding with BLiMP. The Benchmark of Linguistic Minimal Pairs for English (BLiMP) is a 'challenge set' for evaluating the understanding of significant grammatical phenomena in English (Warstadt et al., 2020). BLiMP is composed of 67 distinct datasets, in turn consisting of 1000 minimal pairs, such as 'there was **[*bound* / *unable*]** to be a fish escaping'. For each minimal pair in BLiMP, one sentence is deemed grammatically acceptable, and the other is considered unacceptable. Each set of minimal pairs targets a specific linguistic phenomenon. The 67 phenomena are aggregated further into twelve broader categories. A model evaluated on BLiMP is expected to assign a higher log-likelihood to the acceptable sentence for each minimal pair in the dataset. Its final score is simply the proportion of minimal pairs for which it correctly prefers the acceptable sentence.

5 Results and evaluation

5.1 Main evaluation

This section examines the training trajectories of the four models, providing an overview of their learning processes at a high level. Figure 2 depicts each model's cross-entropy loss, Paloma perplexity and overall BLiMP score throughout training, measured against effective GPU hours.

As Figure 2 illustrates, `pico-relora` performs almost identically to `pico-decoder` on the `tiny` scale, with a more pronounced difference for `small`. This difference is larger for loss than for perplexity. The loss values in the `pico-relora` models show small spikes, coinciding with ReLoRA restarts, which lead to a localised training instability that the models quickly recover from.

`pico-decoder` deviates from the expected perplexity trajectory in the `tiny` run, likely due to the model finding a suboptimal local minimum. `pico-relora` does not experience the same deviation, which may indicate higher training stability, but this can only be confirmed by executing additional training runs. This highlights an important

[1]This version of Dolma is made available on HuggingFace here: `https://huggingface.co/datasets/pico-lm/pretokenized-dolma`.

[2]This pre-processed, subsampled dataset is available here: `https://huggingface.co/datasets/pico-lm/pretokenized-paloma-tinsy`.

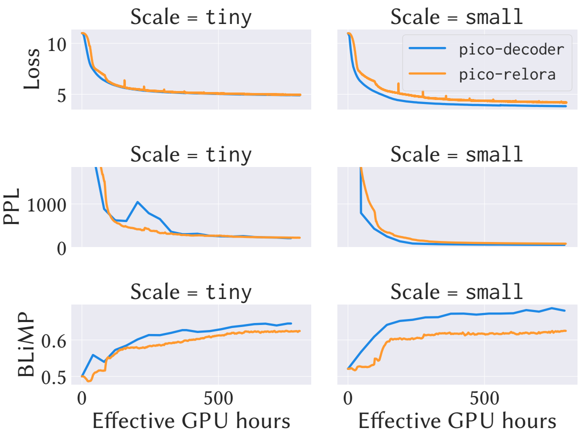

Figure 2: Training trajectories of cross-entropy loss, Paloma perplexity and BLiMP score across the `tiny` and `small` models, plotted against GPU hours taken

limitation of the results throughout this work: only one training run was executed for each model due to time and computational restrictions.

Further to the above findings for loss and perplexity, the `pico-relora` models underperform on BLiMP even more, with pronounced differences at the `small` scale. These results are significant at the 1×10^{-10} level, according to a proportion Z-test set up with an alternative hypothesis that the `pico-decoder` models perform better.

5.2 Analysis

In this section, given `pico-relora`'s underachievement, we strive to identify the inherent properties of the model's training dynamics that may have caused the observed performance drag. To do so, we make use of the 'residual stream' framework (Elhage et al., 2021) through analysis introduced in Diehl Martinez et al. (2024).

5.2.1 Preparation

The 'residual stream' framework for analysing LMs. The 'residual stream' is a mathematical framework for analysing autoregressive decoder-only LMs (Elhage et al., 2021), which focuses on 'residual connections'. The framework considers residual blocks composed of an attention layer and a feed-forward layer, omitting layer normalisation for simplicity. Each of these layers 'reads' from the residual stream with a linear projection and then 'writes' back to it using another linear projection.

Splitting attention head terms into 'Output-Value circuits'. Attention heads can be conceptualised as being composed of two essentially independent calculations: a 'Query-Key' circuit, which calculates the attention pattern, and an 'Output-Value' (OV) circuit, which computes the amount a given token affects output logits (if it is attended to) (Elhage et al., 2021). The OV circuit is calculated by taking the matrix product of the Output and Value matrices. OV circuits, despite being parameterised separately, may be considered as individual, low-rank matrices. We focus on the OV circuit as it 'writes' to the residual stream, a part of the model that has been shown to suffer from performance bottlenecks via the output representations of the model (Godey et al., 2024b).

Learning dynamics & investigated parameters. Following this, learning dynamics analysis is performed on certain activations and parameters of the models' attention and feed-forward layers. Specifically, we choose to analyse activations and parameters that 'write' to the residual stream, namely the OV circuit and SwiGLU W_2 layers (Elhage et al., 2021). Therefore, their behaviour has a salient influence on the evolution of internal representations across layers. This makes them particularly valuable probes for rank bottlenecks and deficiencies.

5.2.2 Proportional effective rank

Given what we know about the rank deficiencies in smaller models, we aim to investigate how they behave and evolve. To achieve this, we utilise PER (Diehl Martinez et al., 2024), a metric that enables size-agnostic comparison of the effective rank of parameter weight matrices. We measure PER on the weight matrices in the feed-forward and attention layers that parameterise the 'writes' back to the residual stream. PER is given by Equation 4 below.

Let the parameter at layer l be $\theta_l \in \mathbb{R}^{d_{\text{model}} \times d_{\text{inter}}}$ where d_{inter} is the dimension of the intermediate representations in either the attention or feed-forward layers.

$$\mathrm{PER}(\theta_l) = \frac{\mathrm{ER}(\theta_l)}{d_{\text{inter}}} \tag{4}$$

where ER (Roy and Vetterli, 2007) is the effective rank metric defined in Equation 5

$$\mathrm{ER}(\theta_l) = \exp\left(-\sum_{k=1}^{Q} \frac{\sigma_k}{\|\boldsymbol{\sigma}\|_1} \log \frac{\sigma_k}{\|\boldsymbol{\sigma}\|_1}\right) \tag{5}$$

where $\boldsymbol{\sigma} = \langle \sigma_1 \ldots \sigma_Q \rangle$ is the vector of singular values of θ_l (in ascending order) and $\|\cdot\|_1$ is the l_1 norm. Stated differently, ER is the entropy computed over the normalised singular values of the weight matrix θ_l.

PER of `pico-relora`'s weights declines throughout training. Figure 3 shows the layer-averaged PER of the OV Circuit and SwiGLU W_2 matrix during training, highlighting clear differences between `pico-relora` and `pico-decoder`. The former generally has a lower PER, with the gap widening as `pico-relora` steadily declines while `pico-decoder` remains flat; except for the `tiny` OV circuit, which shows little change and aligns closely with `pico-decoder`. Aside from this outlier, PER trends are consistent across scales. Confidence intervals for `pico-decoder` are consistently narrow, reflecting low layer-to-layer variability, whereas `pico-relora`'s widen over time.

ReLoRA's gradient updates are rank deficient. Figure 4 shows PER on gradient updates for the Output and Value projections in attention and the SwiGLU W_2 projection. For `pico-decoder`, this is computed directly on gradients, while for `pico-relora`, the LoRA module weights serve as a proxy, since the frozen base matrices receive no gradients at each restart. Observations correspond to the weight measurements: the measured PER values are generally lower for ReLoRA, with this effect exacerbated at the `small` scale, with an outlier for attention at the `tiny` scale.

5.2.3 Condition number

In addition to measured rank deficiencies, we explore further explanations for ReLoRA's underperformance. This leads us to investigate the model's susceptibility to error magnification, for which the condition number (CN or κ) of a matrix (introduced in Equation 6 below) can be a good proxy (Chapra, 2011). The CN measures how sensitive the solution of a linear system is to small changes in the input data or the matrix itself. Furthermore, a higher condition number can be used as a measure of the rounding errors arising from using finite precision arithmetic (Chapra, 2011, p. 208), as is the case when training models. If the κ of a matrix is 10^k, then at least k digits of precision are lost (Cheney et al., 2008, p. 321). The condition number metric is simply the ratio of the largest to the smallest singular values of the input, which indicates how much the output can vary in response to slight variations in the input.

For the ordered vector of singular values $\boldsymbol{\sigma} = \langle \sigma_1 \ldots \sigma_Q \rangle$ of some matrix M, the condition number κ is thus

$$\kappa(M) = \frac{\sigma_Q}{\sigma_1} \tag{6}$$

`pico-relora` has larger CNs in the weight matrices. Figure 5 illustrates the condition numbers of the OV Circuit and SwiGLU W_2 matrices throughout training. The condition number is larger for `pico-relora` SwiGLU matrices, and even more on the `small` scale. For the OV circuit at the `tiny` scale, the κ values are very similar, while at the `small` scale `pico-relora`'s values are greater with a wider confidence interval.

`pico-relora`'s gradient updates are *highly ill-conditioned* early in training. Meanwhile, Figure 6 shows the κ values of the gradient updates of the output and value projections in the attention mechanism and the SwiGLU W_2 matrix. As can be seen, the condition numbers of the ReLoRA model's gradient updates initially start very high, indicating a highly ill-conditioned matrix, and then approach those of the baseline throughout training. These large κ values suggest instability in the gradient updates and vastly increase round-off errors due to fixed precision, by up to eight additional digits of inaccuracy for `pico-relora-small`. For the same run's attention output projection, the condition number spikes at checkpoint step 12,000.

5.2.4 Findings from PER and CN

👎 **Low PER values *drag* ReLoRA's performance down.** Low PER values in the weight matrices and gradient updates exacerbate the rank deficiencies present in smaller models, thereby inhibiting performance compared to full-rank training.

👎 **`pico-relora` is considerably more sensitive to fluctuations in inputs.** The results for CN demonstrate that ReLoRA increases the sensitivity to error magnification caused by small variations in the input. Therefore, even tokens with relatively similar internal representations can have vastly different outputs.

This behaviour is compounded by the *anisotropy of SLMs*, which refers to the phenomenon where the internal representations of tokens are *unevenly distributed* and *highly clustered* within the representational space (Godey et al., 2024a). As a result, even small changes in input can cause a token embedding to shift into a sparse or differently clustered region, thereby amplifying differences in the model's output.

When both these effects combine, they lead to a model which is highly sensitive to any minor fluctuations in its inputs.

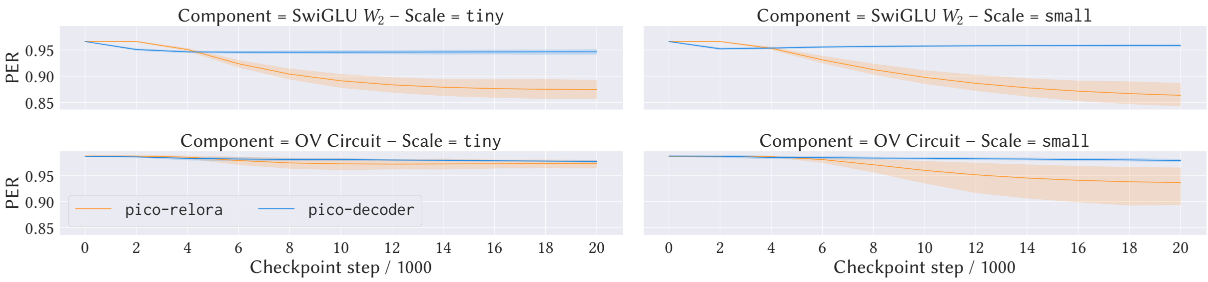

Figure 3: Proportional effective rank (PER) values of the parameters of the OV Circuit and the SwiGLU W_2 matrix, averaged over the models' layers. Values are shown with the 95% confidence interval.

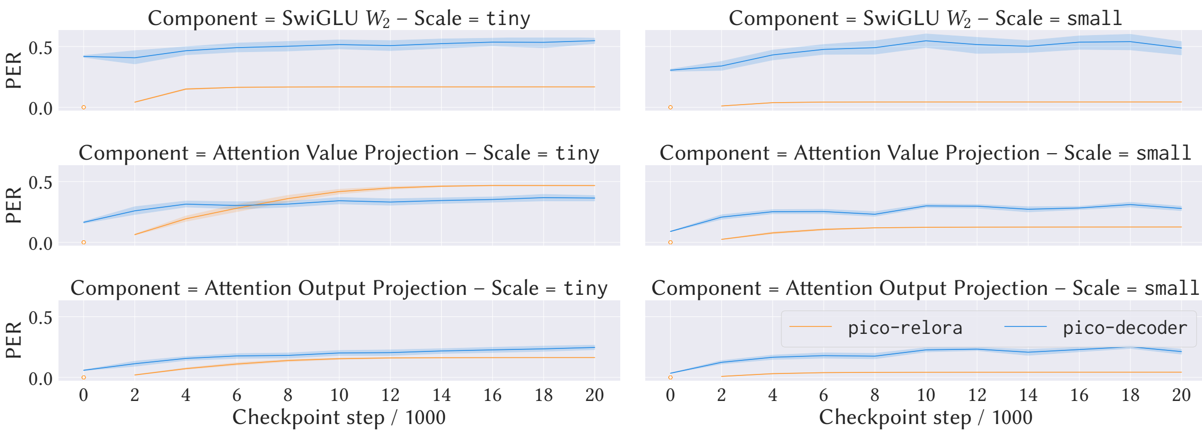

Figure 4: Proportional effective rank (PER) values of the gradient updates of the output and value projections in the attention mechanism and the SwiGLU W_2 matrix, averaged over the models' layers. Values are shown with bands representing the 95% confidence interval. The hollow circles represent `NaN` values.

6 Conclusions

We present the first systematic investigation of ReLoRA for SLM pretraining, evaluating models at 11M (`tiny`) and 66M (`small`) parameters across various efficiency and performance metrics.

6.1 Findings

We identify three key findings:

(1) 👎 **Drag: ReLoRA degrades pretraining performance in SLMs.** Across loss, Paloma perplexity, and BLiMP evaluation, ReLoRA consistently underperforms conventional full-rank training. The performance gap is minor for `tiny` models but grows substantially as model size increases to the `small` scale. However, this might be caused by minimal parameter differences between the two tiny models, leading to a surprising performance loss (compared to the findings of (Lialin et al., 2023) on large models) as model scale increases. One possible explanation is that the tiny model's parameterisation leaves little difference between ReLoRA and full-rank updates, therefore leading to a lower-than-expected drop in performance.

(2) 👎 **Drag: ReLoRA exacerbates low-rank bottlenecks and training instability.** Learning dynamics analysis indicates that ReLoRA leads to reduced PERs of the models' parameters and gradient updates. Alongside this, ReLoRA induces highly ill-conditioned gradient updates that increase the models' susceptibility to numerical errors. Both of these exacerbate existing issues caused by the inherent anisotropy of SLM representations.

(3) **Parameter-efficient pretraining does not trivially extend from large to small models.** Unlike in large LMs, where low-rank updates can still capture rich training signals, SLMs appear to have insufficient redundancy in their representations, resulting in greater sensitivity to ReLoRA's repeated low-rank projections.

6.2 Implications

This paper highlights a principal limitation of low-rank pretraining for SLMs, suggesting that for a

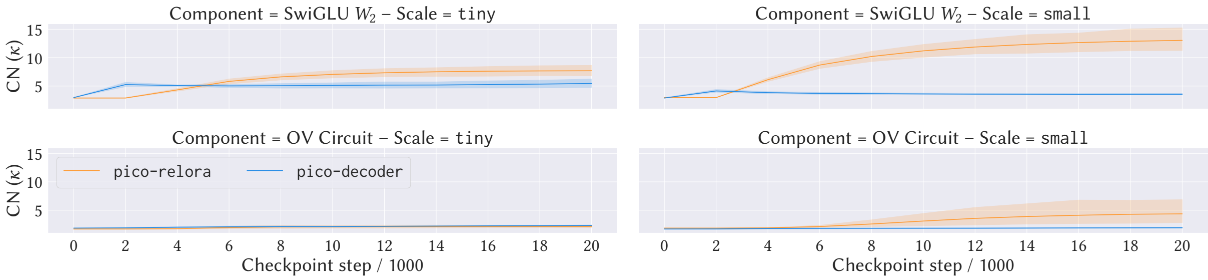

Figure 5: Condition numbers of the weight matrices of the OV Circuit and the SwiGLU W_2 matrix, averaged over the models' layers. Values are shown with the 95% confidence interval.

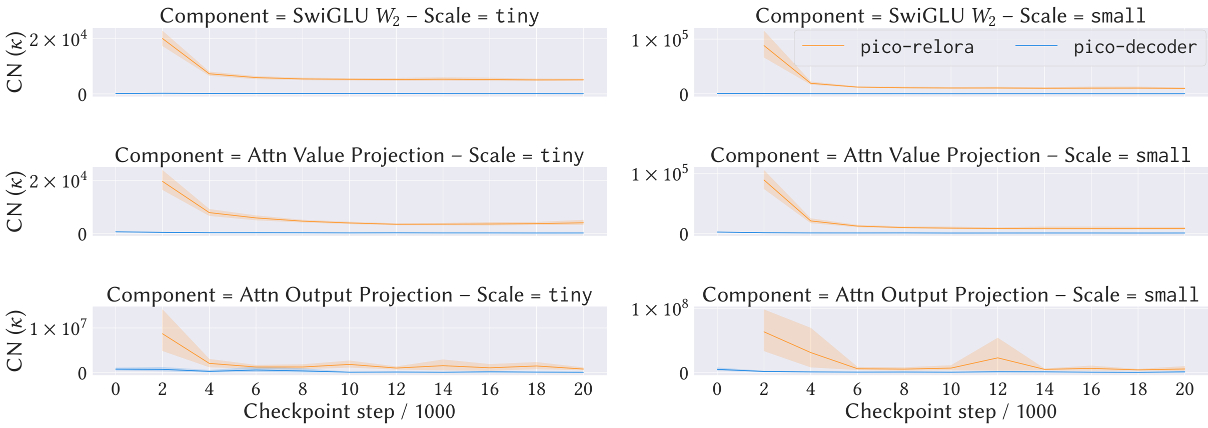

Figure 6: Condition numbers of the gradient updates of the output and value projections in the attention mechanism and the SwiGLU W_2 matrix, averaged over the models' layers. Values are shown with bands representing the 95% confidence interval. The values at the initial checkpoints are omitted since they are zero matrices, with undefined κ.

parameter-efficient training method to succeed in the low-compute domain, it likely must preserve higher-rank update spaces or adapt to models' intrinsic dimensionalities. Furthermore, our results prompt reflection on whether parameter-efficient training methods are necessary for SLMs at all. Unlike SOTA multi-billion-parameter models, SLMs can be trained on a single modern GPU (and often even on consumer hardware), raising the question of whether parameter-efficiency provides any benefits in this regime.

While these are negative results, they are informative in highlighting the boundaries at which parameter-efficient pretraining breaks down. Our study thus highlights where new methods are most needed.

Future investigations should therefore focus on novel hybrid approaches. For instance, coupling low-rank adapters with selective full-rank updates or utilising dynamic rank adaptation techniques (such as DyLoRA (Valipour et al., 2023)) to minimise representational losses. Our results aim to inform the design of future low-resource LM training algorithms, emphasising that efficiency gains cannot come at the cost of substantial expressivity and performance loss.

Limitations

While our work offers valuable contributions, there are a few minor considerations to keep in mind:

(1) Single-run experiments. Due to constraints on computational resources, we performed only one training run per configuration. While the results are interesting, additional seeds at each scale would lead to more robust results.

(2) Limited model scale and diversity. Likewise, the experiments were performed at only two scales and evaluated with a single decoder-only architecture. Investigating other intermediate scales and encoder-decoder models may provide further insight and more robust trends.

(3) Limited hyperparameter exploration. ReLoRA involves several hyperparameters, such as restart frequency, LoRA rank, and optimiser state pruning ratio. Our study evaluates a single shared configuration and omits the full-rank warm start included by Lialin et al. (2023), which may not reflect the method's ideal performance.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. 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. Association for Computational Linguistics.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. 2023. Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets. *Preprint*, arXiv:2311.15127.

Enric Boix-Adsera, Etai Littwin, Emmanuel Abbe, Samy Bengio, and Joshua Susskind. 2023. Transformers learn through gradual rank increase. *Advances in Neural Information Processing Systems*, 36:24519–24551.

Steven Chapra. 2011. *EBOOK: Applied Numerical Methods with MATLAB for Engineers and Scientists*. McGraw Hill.

Elliott W. Cheney, Elliott W. Cheney, and David Kincaid. 2008. *Numerical Mathematics and Computing*, 6. ed., internat. student ed edition. Thomson Brooks/Cole, Belmont, Calif.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, and 48 others. 2023. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Tri Dao. 2023. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. *Preprint*, arXiv:2307.08691.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. *Advances in Neural Information Processing Systems*, 36:10088–10115.

Richard Diehl Martinez. 2025. Pico: A lightweight framework for studying language model learning dynamics.

Richard Diehl Martinez, Pietro Lesci, and Paula Buttery. 2024. Tending Towards Stability: Convergence Challenges in Small Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3275–3286, Miami, Florida, USA. Association for Computational Linguistics.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, and 1 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12.

Allyson Ettinger. 2020. What BERT Is Not: Lessons from a New Suite of Psycholinguistic Diagnostics for Language Models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Vlad Fomenko, Han Yu, Jongho Lee, Stanley Hsieh, and Weizhu Chen. 2024. A Note on LoRA. *Preprint*, arXiv:2404.05086.

Nathan Godey, Éric Clergerie, and Benoît Sagot. 2024a. Anisotropy Is Inherent to Self-Attention in Transformers. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 35–48, St. Julian's, Malta. Association for Computational Linguistics.

Nathan Godey, Éric Villemonte de la Clergerie, and Benoît Sagot. 2024b. Why do small language models underperform? Studying Language Model Saturation via the Softmax Bottleneck. In *First Conference on Language Modeling*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, and 24 others. 2024. OLMo: Accelerating the Science of Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809, Bangkok, Thailand. Association for Computational Linguistics.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *Preprint*, arXiv:2001.08361.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. 2023. ReLoRA: High-Rank Training Through Low-Rank Updates. In *The Twelfth International Conference on Learning Representations*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *Preprint*, arXiv:1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. *Preprint*, arXiv:1711.05101.

Ian Magnusson, Akshita Bhagia, Valentin Hofmann, Luca Soldaini, Ananya Harsh Jha, Oyvind Tafjord, Dustin Schwenk, Evan Walsh, Yanai Elazar, Kyle Lo, and 1 others. 2024. Paloma: A benchmark for evaluating language model fit. *Advances in Neural Information Processing Systems*, 37:64338–64376.

Sebastian J. Mielke, Ryan Cotterell, Kyle Gorman, Brian Roark, and Jason Eisner. 2019. What Kind of Language Is Hard to Language-Model? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4975–4989, Florence, Italy. Association for Computational Linguistics.

Jacob Morrison, Clara Na, Jared Fernandez, Tim Dettmers, Emma Strubell, and Jesse Dodge. 2024. Holistically Evaluating the Environmental Impact of Creating Language Models. In *The Thirteenth International Conference on Learning Representations*.

Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. 2022. Signal Propagation in Transformers: Theoretical Perspectives and the Role of Rank Collapse. *Advances in Neural Information Processing Systems*, 35:27198–27211.

OpenAI. 2024. GPT-4o System Card. *Preprint*, arXiv:2410.21276.

Olivier Roy and Martin Vetterli. 2007. The effective rank: A measure of effective dimensionality. In *2007 15th European Signal Processing Conference*, pages 606–610.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green AI. *Communications of the ACM*, 63(12):54–63.

Noam Shazeer. 2020. GLU Variants Improve Transformer. *Preprint*, arXiv:2002.05202.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, and 17 others. 2024. Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15725–15788, Bangkok, Thailand. Association for Computational Linguistics.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. RoFormer: Enhanced Transformer with Rotary Position Embedding. *Preprint*, arXiv:2104.09864.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. DyLoRA: Parameter-Efficient Tuning of Pre-trained Models using Dynamic Search-Free Low-Rank Adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287, Dubrovnik, Croatia. Association for Computational Linguistics.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The Benchmark of Linguistic Minimal Pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Biao Zhang and Rico Sennrich. 2019. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

A The `pico-decoder` model

`pico-decoder` (Diehl Martinez, 2025) is a Llama-based (Touvron et al., 2023) model, which forms the base of the ablation study investigated in this work. It is illustrated in Figure 7.

`pico-decoder` includes the following features and optimisations:

- Root-Mean-Square Layer Normalisation (Zhang and Sennrich, 2019)
- RoPE embeddings (Su et al., 2023)
- Grouped-query attention (Ainslie et al., 2023) with key-value caching

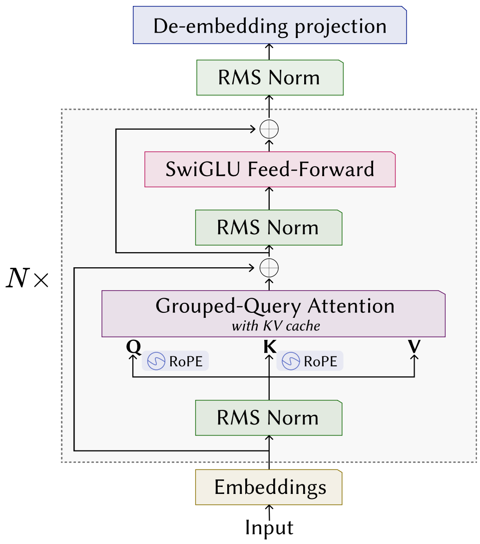

Figure 7: `pico-decoder`, a Llama-based model

- FlashAttention-2 (Dao, 2023)
- SwiGLU activations (Shazeer, 2020)

B Training infrastructure and runtimes

The `pico-relora` training runs were conducted on a single Ampere GPU node (four GPUs) each, hosted by CSD3. By contrast, the `pico-decoder` runs were performed on four Ampere nodes (sixteen GPUs).

The Ampere nodes are Dell PowerEdge XE8545 servers consisting of two AMD EPYC 7763 64-Core Processors 1.8 GHz, 1000 GiB RAM, four NVIDIA A100-SXM-80GB GPUs, and dual-rail Mellanox HDR200 InfiniBand interconnect. Each A100 GPU is made up of 6912 FP32 CUDA Cores.[3]

The wall-clock runtime and total GPU hours of each of the runs are shown in Table 3. For the `tiny` models, `pico-relora` takes 13.5% fewer GPU hours to train, while for the `small` scale, it was 14.7% more efficient. These are not entirely equivalent comparisons, however, as the `pico-relora` runs were configured to checkpoint and evaluate more frequently and with more gradient accumulation steps, alongside other minor differences in the runtime environment.

Model	Wall-clock time / h	GPU hours
`t-dec`	52.96	847.37
`s-dec`	61.29	980.56
`t-rel`	183.17	732.70
`s-rel`	215.87	863.47

Table 3: Wallclock time and equivalent GPU hours for each of the four training runs. `t` = `tiny`, `s` = `small`, `dec` = `decoder`, `rel` = `relora`.

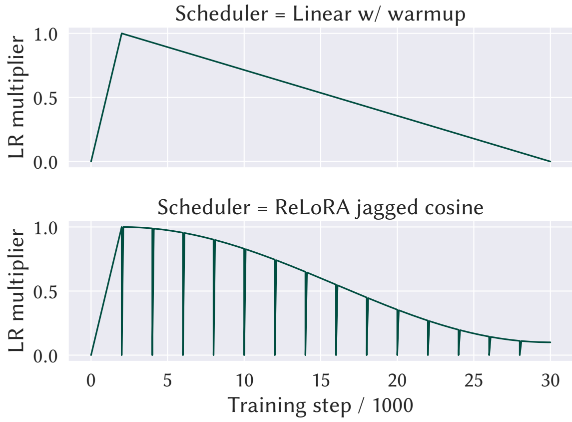

Figure 8: Graph showing the linear LR schedule used by `pico-decoder` and ReLoRA's jagged cosine scheduler (Lialin et al., 2023). Both are shown here, configured with a total of 30,000 steps and an initial warmup of 2000 steps. The ReLoRA scheduler is additionally configured with 100 post-restart warmup steps and a reset frequency of 2000.

C Full hyperparameter configurations

C.1 Learning rate scheduler and optimiser configuration – *shared*

Table 4 depicts the learning rate and optimisation configurations for both runs. It has an initial linear warmup of 2000 steps, and resets in line with the ReLoRA frequency specified in Table 2. After each reset, the learning rate is linearly re-warmed over 100 steps before the standard cosine decay is resumed. The learning rate decays down to a minimum of 10% of the initial rate. The baseline runs use the linear scheduler configured with a 2500 step initial warmup. However, as this is indivisible by the 2000 step ReLoRA frequency, the jagged cosine scheduler uses the nearest multiple, a 2000 step initial warmup. Both schedulers are shown in Figure 8.

[3] This information is sourced here: https://docs.hpc.cam.ac.uk/hpc/user-guide/a100.html.

Parameter	Value
Optimiser	adamw
Learning rate	3×10^{-4}
LR scheduler	relora_jagged_cosine
LR warmup steps	2000
Min LR ratio	0.1
Restart warmup steps	100

Table 4: Learning rate scheduler and optimiser configuration for training runs

C.2 Data configuration – *shared*

Table 5 configures the data pipeline for the training runs. The training and perplexity datasets are `pretokenized-dolma` and `pretokenized-paloma-tinsy`, as described in Section 4.2. The tokeniser used, `allenai/OLMo-7B-0724-hf`, is the one used for the OLMo model[4] (Groeneveld et al., 2024).

Parameter	Value
Vocab size	50,304
Batch size	1024
Max seq length	2048

Table 5: Dataset configuration

C.3 Model configuration – *shared*

Table 6 depicts the model configuration parameters for the `tiny` and `small` training runs. The two runs differ only in their d_{model} and d_{ff} parameters, keeping the depth of the model and attention architecture constant.

Parameter	`tiny`	`small`
n_{layers}	12	
n_{heads}	12	
$n_{\text{heads}_{\text{kv}}}$	4	
d_{model}	96	384
d_{ff}	384	1536

Table 6: Model configuration for training runs

[4]Which can be found on HuggingFace here: https://huggingface.co/allenai/OLMo-7B-0724-hf.