LaX: Boosting Low-Rank Training of Foundation Models via Latent Crossing

Ruijie Zhang[*], **Ziyue Liu**[*], **Zhengyang Wang**, **Zheng Zhang**[†]
University of California at Santa Barbara
{ruijiezhang, ziyueliu, zhengyangwang}@ucsb.edu, zhengzhang@ece.ucsb.edu

Abstract

Training foundation models such as ViTs and LLMs requires tremendous computing cost. Low-rank matrix or tensor factorization offers a parameter-efficient alternative, but often downgrades performance due to the restricted parameter space. In this work, we introduce **Latent Crossing (LaX)** – a simple yet effective plug-and-play module that enhances the capacity of low-rank models by enabling information flow across low-rank subspaces. We extensively validate the benefits of LaX on pre-training tasks with ViT-Base/Large and LLaMA-like models ranging from 60M to 1B parameters. LaX boosts low-rank model performance to match or exceed the full-rank baselines while using 2-3$\times$ fewer parameters. When equipped with low-rank adapters (i.e., LoRA [23]) for fine-tuning LLaMA-7/13B, LaX consistently improves performance on arithmetic and common sense reasoning tasks with negligible cost.

1 Introduction

Following neural scaling laws [28, 22, 33], the size and training data of foundation models have grown rapidly, exemplified by models such as ViT-22B [7], GPT-3 (175B) [3], LLaMA-3 (405B) [13], and PaLM (504B) [5]. These large-scale foundation models have achieved remarkable success in diverse applications such as language and vision. However, their success comes at immense computing cost, typically on the scale of multi-million GPU hours per pre-training run. As the unsustainable trend continues, training or even deploying such foundation models has become prohibitively expensive for most research institutions and organizations around the world.

To address these challenges, the community has become increasingly interested in low-rank approximation techniques. This is largely motivated by the empirical observation that weight matrices in deep neural networks often exhibit low effective ranks [1, 11, 27, 64, 43]. Classical matrix compression techniques (such as singular value decomposition (SVD) [10]) and tensor decomposition methods (e.g. Canonical Polyadic (CP), Tensor Train (TT) [16, 4, 29] and Tucker decomposition [51]) have been widely applied to reduce the number of trainable parameters by instantiating and updating the lightweight low-rank factors [31, 45, 12, 57, 60, 56, 39, 65, 34]. These approaches exemplify the paradigm of "low-rank training" and have achieved varying degrees of success. In particular, parameter-efficient fine-tuning (PEFT) [23, 63, 41, 17, 62] has drastically reduced the barrier to fine-tuning large language models while producing competitive results. Recent efforts [38, 66, 15, 60] have extended similar concepts to pre-training. Although low-rank methods typically reduce the model size and computing cost, they introduce a critical trade-off: smaller ranks yield lower capacity and often harm performance, whereas larger ranks incur additional cost, undermining the intended efficiency (see Fig. 1 (a)).

In this work, we propose **Latent Crossing** (LaX), a lightweight, drop-in module designed to enhance the capacity of low-rank models without explicitly increasing matrix/tensor ranks. By allowing information flow across low-rank subspaces via residual connections, LaX improves model performance

Preprint. Under review.

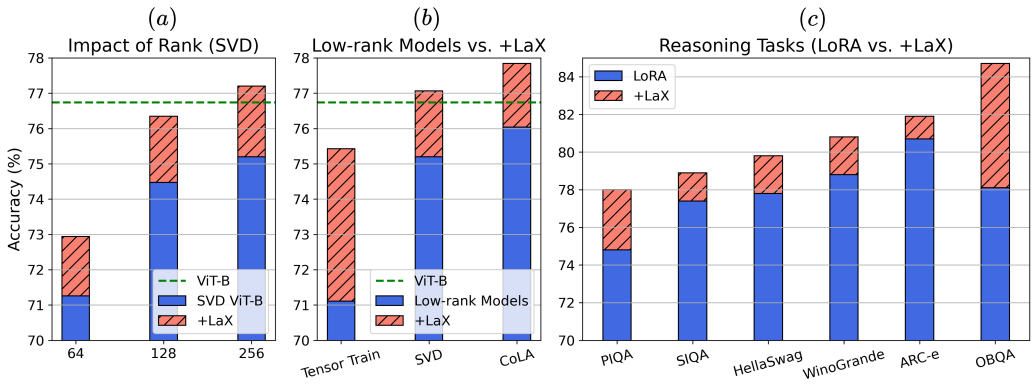

Figure 1: LaX boosts the performance of low-rank training methods. (a) SVD-based pre-training ViT-B on ImageNet-1K with different matrix ranks: lower-rank leads to greater performance drop; LaX consistently improves the performance in all settings. (b) Pre-training ViT-B on ImageNet-1K with different low-rank methods. LaX significantly improves performance for all low-rank methods, even surpassing the full-rank pre-training. (c) Fine-tuning LLaMA-7B on commonsense reasoning tasks using LoRA, with and without LaX respectively. LaX improves LoRA's fine-tuning performance in all tasks.

while keeping the parameter budget nearly unchanged. Importantly, LaX can be seamlessly integrated with existing low-rank modules such as LoRA [23], SVD, CoLA [43] and TT [45], serving as a **plug-and-play performance booster** that significantly narrows or eventually closes the gap between low-rank and full-rank models.[1]

We summarize our contributions as follows:

1. We propose **LaX**, a lightweight module that increases the capacity of existing low-rank structures without compromising efficiency. By allowing information flow across low-rank subspaces via residual connections, LaX consistently boosts performance in both pretraining and fine-tuning settings (Fig. 1).
2. We design **LaX Gate** to align mismatched bottlenecks for low-rank models. To support diverse architectural and computational constraints, we introduce several variants of the LaX gating mechanism, each balancing expressiveness and efficiency under different deployment settings. We also provide practical guidelines for adapting LaX gating to a variety of tasks.
3. We evaluate LaX in a set of low-rank pre-training and fine-tuning experiments for both language and vision foundation models. In ViT pre-training, LaX improves accuracy by up to $\mathbf{4.32}\%$ on ImageNet-1K. For LLM pre-training, LaX shows **consistent gains** across various model scales and different low-rank architectures. When combined with LoRA for fine-tuning, LaX enhances reasoning capabilities of LLaMA-7B/13B on both arithmetic and commonsense reasoning tasks.

2 Related Works

2.1 Low-rank Factorization for Neural Networks

To mitigate the high computational and storage costs associated with large models, low-rank factorizations have been widely explored as an effective strategy [47, 1]. Early efforts focused on applying low-rank matrix factorization, such as SVD, to compress layers [8, 26, 31, 30, 61]. More recently, LoRA-style adapters [23, 63, 41, 17, 62] extend this idea by adapting SVD-like modules onto frozen pre-trained weights, enabling efficient fine-tuning of large foundation models. Besides, low-rank tensor factorization, including tensor train (TT) [46, 6, 54], Canonical Polyadic (CP) [16, 4, 29], Tucker decomposition [51], and other tensor-based formats [67, 32, 2, 50] have shown promise for reducing complexity of models [40, 36, 52, 42, 39, 65, 34, 60]. In this paper, we focus on representative methods from both directions:

Low-rank Matrix Factorization. SVD factorizes a weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ as $\mathbf{W} = \mathbf{BA}$, where $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}$, resulting in a reduced parameter count of $r(d_{\text{in}} + d_{\text{out}})$.

[1]We provide our code here

CoLA [43] further extends this factorization to an autoencoder by injecting a nonlinear activation σ between $\mathbf{A}$ and $\mathbf{B}$, replacing a linear layer $\mathbf{W}\mathbf{x}$ with the bottleneck structure $\mathbf{B}\,\sigma(\mathbf{A}\mathbf{x})$.

Low-rank Tensor Factorization. We adopt tensor train decomposition [46] as a representative higher-order factorization method. It reshapes a weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ into an order-n tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{d_0 \times d_2 \cdots \times d_{n-1}}$ with $d_{\text{out}} \times d_{\text{in}} = \prod_{i=0}^{n-1} d_i$, and decomposes it with a sequence of tensor cores $\{\boldsymbol{\mathcal{C}}^0, \boldsymbol{\mathcal{C}}^1, \ldots, \boldsymbol{\mathcal{C}}^{n-1}\}$, where each $\boldsymbol{\mathcal{C}}^i \in \mathbb{R}^{r_i \times d_i \times r_{i+1}}$. The weight is represented via a chain of tensor contractions:

$$\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{C}}^0 \times_{3,1} \boldsymbol{\mathcal{C}}^1 \times_{3,1} \cdots \times_{3,1} \boldsymbol{\mathcal{C}}^{n-1}.$$

However, the extent of parameter reduction, as well as its benefits heavily depend on the chosen rank. Lower-rank settings often suffer from a loss of expressiveness and degrade model performance [66, 43, 23, 59, 47, 1], while higher ranks reintroduce computational overhead. In this work, we propose a simple, plug-and-play module that complements general low-rank training methods in neural networks, aiming to **recover lost performance without compromising their efficiency**.

2.2 Residual Mechanism

ResNet [18] stands as one of the most influential milestones in deep learning by introducing a skip connection that routes a layer's input directly to its output. This *residual mechanism* mitigates the vanishing gradients and enables the stable training of deep networks with hundreds of layers. This design principle has been broadly adopted in numerous architectures, including recurrent neural networks [14], transformers [9, 53], and diffusion-based models [21]. In addition to its empirical success, the authors provided a theoretical justification for the residual connection [19]. Building on this foundation, subsequent research has proposed various improvements and theoretical analyses to further enhance the residual learning paradigm [58, 20, 55, 35], consistently emphasizing the central role of residual pathways in improving both convergence and generalization in deep networks.

Inspired by ResNet, we propose **Latent Crossing (LaX)**. LaX serves as a model performance booster by enabling information flow across low-rank subspaces, restoring expressiveness often lost due to rank constraints. Across a wide range of tasks, LaX delivers consistent performance gains while preserving the efficiency advantages of low-rank architectures.

3 The LaX Method

Our goal is to augment existing low-rank models with a lightweight module that recovers the performance typically lost due to the low-rank constraints. In Section 3.1, we present the background and motivation behind this work. Following this, Section 3.2 introduces the design of the LaX module for different low-rank structures, Section 3.3 introduces LaX Gate and outlines its key variants, and Section 3.5 offers practical guidelines to facilitate its integration into a wide range of low-rank training frameworks.

3.1 Latent Crossing

As discussed in Section 2.1, given a weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ from an arbitrary linear layer, low-rank methods approximate it as low-rank factors. We take SVD as a motivating example to illustrate this concept. Let $\mathbf{x}_i \in \mathbb{R}^{d_{\text{in}}}$ denote the input to the i-th low-rank layer. A down-projection matrix $\mathbf{A}_i \in \mathbb{R}^{r \times d_{\text{in}}}$ maps $\mathbf{x}_i$ into a lower-dimensional latent representation $\mathbf{h}_i \in \mathbb{R}^r$, which is subsequently transformed back to the output space using an up-projection matrix $\mathbf{B}_i \in \mathbb{R}^{d_{\text{out}} \times r}$:

$$\mathbf{h}_i = \mathbf{A}_i \mathbf{x}_i \in \mathbb{R}^r, \quad \mathbf{y}_i = \mathbf{B}\big(\mathbf{A}_i \mathbf{x}_i\big) = \mathbf{B}_i \mathbf{h}_i \in \mathbb{R}^{d_{\text{out}}}. \tag{1}$$

This factorization reduces the number of parameters by choosing a smaller rank r and compresses input into a narrow latent space, which can lead to information bottlenecks, often resulting in a drop of performance due to the constrained searching space. Empirically, increasing the rank r typically improves performance but diminishes the efficiency benefits of the low-rank approach due to increased parameter count and computation.

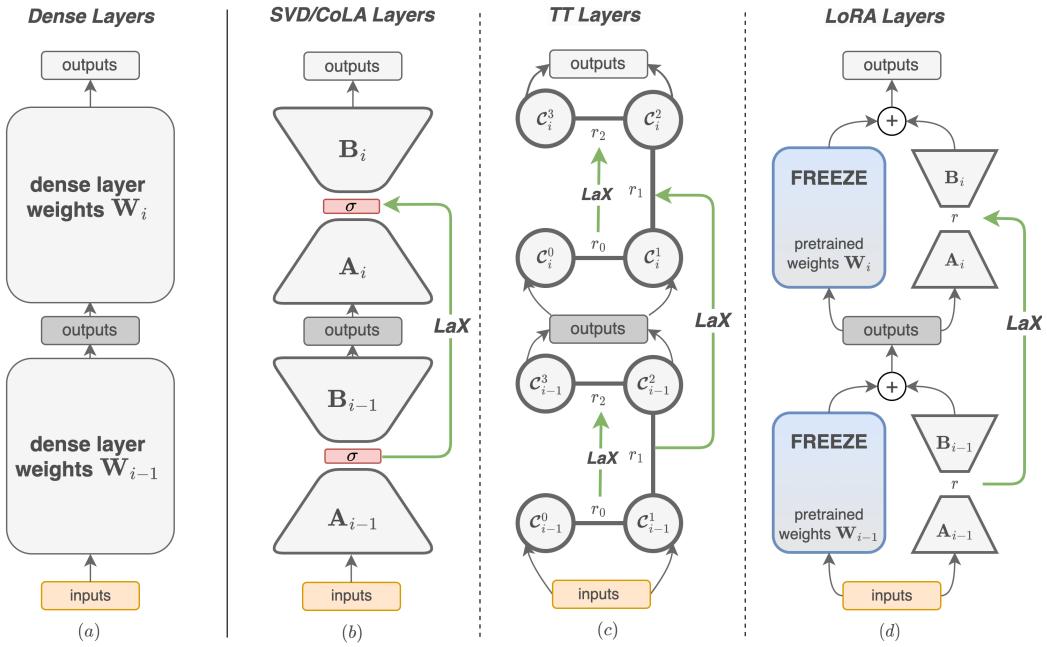

Figure 2: LaX is a general module that can be plugged into low-rank neural network models. **(a)** Dense layers: full information flow, effective but computationally expensive. **(b)** SVD/CoLA[43] layers: rank-r bottlenecks with two factors; LaX can be inserted into the latent space between layers. **(c)** Tensor-train layers: bottleneck structure with four tensor cores, where data flow is governed by tensor contractions; LaX can be applied either between cores or across layers. **(d)** LoRA adapters: LaX can be placed between different adapters.

Our motivation is that, can we **improve the performance of low-rank layers without explicitly increasing the physical rank r?** Our answer is yes. Instead of directly applying the up-projection $\mathbf{B}_i$ to $\mathbf{h}_i$, LaX incorporates latent features from the previous layer into the up-projection process. Formally, we propose LaX as follows:

$$\begin{aligned} \mathbf{h}_{i-1} = \mathbf{A}_{i-1}\mathbf{x}_{i-1}, \quad \mathbf{h}_i = \mathbf{A}_i\mathbf{x}_i \in \mathbb{R}^r, \\ \tilde{\mathbf{y}}_i = \mathbf{B}_i(\mathbf{h}_i + \mathbf{h}_{i-1}) \in \mathbb{R}^{d_{\text{out}}}. \end{aligned} \tag{2}$$

Equivalently, if we stack inputs as $\tilde{\mathbf{x}}_i := \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_{i-1} \end{bmatrix} \in \mathbb{R}^{2d_{\text{in}}}$, then LaX can be formulated as

$$\tilde{\mathbf{y}}_i = \mathbf{W}_i^{(LaX)}\tilde{\mathbf{x}}_i, \qquad \boxed{\mathbf{W}_i^{(LaX)} := \begin{bmatrix} \mathbf{B}_i\mathbf{A}_i & \mathbf{B}_i\mathbf{A}_{i-1} \end{bmatrix} \in \mathbb{R}^{d_{\text{out}} \times 2d_{\text{in}}}.} \tag{3}$$

Since $\mathbf{h}_{i-1}$ is naturally produced by the preceding layer during the forward pass, this implicit reuse of intermediate representations facilitates direct information flow across consecutive low-rank projections, requiring no additional parameters or computation overhead.

3.2 Variants of LaX

LaX is a general module that is widely applicable to low-rank structures. Eq (2) mainly describes its implementation on matrix factorization methods, where LaX is applied across two consecutive layers. We also refer to this implementation as **Inter-Layer LaX**. In modern architectures such as the transformer, we apply Inter-Layer LaX between the same type of layers across transformer blocks, i.e., from attention (QKV projection) to attention, and from MLP to MLP. This design preserves structural and semantic alignment in residuals while avoiding cross-type interference.

For more fine-grained low-rank structure, such as tensor factorization methods, LaX is not limited to cross-layer only. Take the tensor-train representation in Fig. 3 as an example: $\mathbf{W}$ (dropping layer index for simplicity) is approximated by 6 low-rank factors $\{\mathcal{C}^0, \mathcal{C}^1, \ldots, \mathcal{C}^5\}$, therefore, a series of latent features will be produced when sequentially contracting each factor, such as $\mathcal{C}^0\mathbf{x}$, $\mathcal{C}^0\mathcal{C}^1\mathbf{x}$, $\mathcal{C}^0\mathcal{C}^1\mathcal{C}^2\mathbf{x}$, etc. Along this contraction sequence, earlier results can be used as residuals to form multiple LaX pathways. We refer to this implementation as **Intra-Layer LaX**.

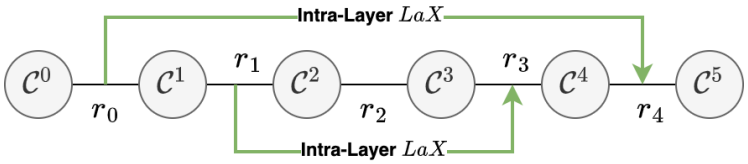

Figure 3: A 6-core Tensor Train layer with the symmetric setting. For Tensor Train layers with identical input and output shapes, we can naturally arrange the tensor ranks in a symmetric configuration, where $r_0 = r_4$ and $r_1 = r_3$ in this example. This reduces the need for shape transformation operations, making **Intra-Layer** LaX more efficient when applied.

3.3 LaX Gates

When the latent dimensions are aligned (e.g., SVD with the same rank between layers, symmetric TT ranks within a layer), direct residual pathways can be formed without introducing extra parameters. For example, in QKV projection layers with symmetric TT setup (Fig.3), we configure to ensure residual compatibility. However, a direct addition becomes infeasible when latent features have mismatched dimensions. To handle this, we introduce LaX Gate, a module that aligns and modulates latent features before residual fusion (Fig. 4). The gated residual formulation becomes:

$$\tilde{\mathbf{y}}_i = \mathbf{W}_i^{(LaX)}\,\tilde{\mathbf{x}}_i, \qquad \mathbf{W}_i^{(LaX)} := \big[\,\mathbf{B}_i\mathbf{A}_i \quad \mathbf{B}_i G_i \mathbf{A}_{i-1}\big]. \tag{4}$$

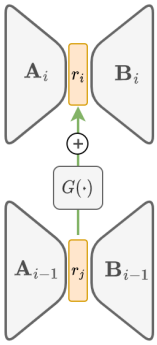

Figure 4: LaX Gate.

To accommodate different architectural needs and promote the versatility of LaX, we unify the notion by introducing the following Gate variants:

- **Identity Gate**: Passes latent features through a direct addition without introducing additional parameters, i.e. $G(\cdot) = 1$.
- **Linear Gate**: Introduces a single trainable parameter to control how much information is passed forward, i.e. $G(\cdot) = \beta$, where $\beta \in \mathbb{R}$ is a trainable parameter.
- **Tensor Gate**: As illustrated in Fig. 5, this variant first folds the latent feature vector into a tensor $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{r_0 \times 1 \times r_1}$, then contracts it with two learnable gate tensor cores: $\boldsymbol{\mathcal{C}}^0 \in \mathbb{R}^{1 \times r_0^{'} \times r_0}$ and $\boldsymbol{\mathcal{C}}^1 \in \mathbb{R}^{r_1 \times r_1^{'} \times 1}$,i.e. $\boldsymbol{\mathcal{R}}^{'} = \boldsymbol{\mathcal{C}}^0 \times_{3,1} \boldsymbol{\mathcal{R}} \times_{3,1} \boldsymbol{\mathcal{C}}^1 \times_{3,1} \in \mathbb{R}^{r_0^{'} \times 1 \times r_1^{'}}$ to match targeting shape $r_0^{'} \times r_1^{'}$. In the two-core setting, each gating core includes a singleton dimension. This dimension can be generalized to larger sizes when extending the design to more than two gating cores, allowing for greater flexibility across different use cases.
- **Dense Gate**: Passes latent feature using a $G \in \mathbb{R}^{r \times r}$ linear layer.

We remark that the overhead introduced by LaX in terms of parameter count and computation is often minimal, since the latent rank r is relatively small (e.g., 64 or 128) in low-rank models. In addition to addressing dimensional misalignment, we empirically find that LaX Gate can further boost performance with negligible parameter overhead (see details in Section 4.1 for ViT pre-training with Tensor Gate). Therefore we also experiment LaX Gate on scenarios where dimensions are matched.

3.4 Feature Normalization

Additionally, following the common practice of normalizing features after a residual connection, we postpend a *Layer Normalization* (LN) to each LaX pathway. With this normalization, the final form of LaX is as follows:

$$\tilde{\mathbf{y}}_i = \mathrm{LN}(\mathbf{W}_i^{(LaX)}\,\tilde{\mathbf{x}}_i), \qquad \mathbf{W}_i^{(LaX)} := \big[\,\mathbf{B}_i\mathbf{A}_i \quad \mathbf{B}_i G_{i-1} \mathbf{A}_{i-1}\big]. \tag{5}$$

3.5 Practical Guideline

Here, we provide practical guidelines for selecting the appropriate LaX variant based on empirical observations across different tasks:

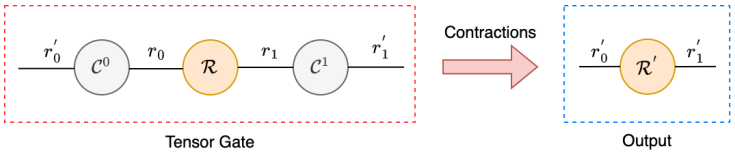

Figure 5: Two-Core Tensor Gate. A residual tensor $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{r_0 \times 1 \times r_1}$ is contracted with two gating tensor cores, $\boldsymbol{\mathcal{C}}^0$ and $\boldsymbol{\mathcal{C}}^1$, producing a transformed residual tensor $\boldsymbol{\mathcal{R}}^{'} \in \mathbb{R}^{r_0^{'} \times 1 \times r_1^{'}}$.

		ViT-B		ViT-L	
Method	Variant	# Params (M)	Accuracy (%)	# Params (M)	Accuracy (%)
Original	-	86.56	76.74	304.33	77.10
SVD	Base Model	44.17	75.20	115.77	76.81
	+ LaX (Ours)	44.24	**77.20** (+2.00)	115.92	**78.60** (+1.79)
Tensor Train	Base Model	41.18	71.11	101.97	75.21
	+ LaX (Ours)	41.44	**75.43** (+4.32)	102.10	**77.77** (+2.56)
CoLA	Base Model	44.17	76.04	115.77	77.63
	+ LaX (Ours)	44.24	**77.84** (+1.80)	115.92	**79.07** (+1.44)

Table 1: Accuracy comparison of pre-training on ImageNet-1k datasets. LaX consistently improves pre-training performance across various low-rank models and scales. When applied to CoLA [43], CoLA+LaX achieves the highest accuracy on both ViT-B and ViT-L. Tensor Train models observe the largest gains, with improvements of +4.32%/+2.56% on ViT-B/L.

- **ViTs Pre-training Task**: Vision Transformer pre-training typically involves multiple epochs over medium-sized datasets. Under this setting, we observe (see Tab. 2) that the **Tensor Gate** consistently outperforms other gate variants. We therefore recommend using the **Tensor Gate** for ViT pre-training tasks.
- **LLMs Pre-training Task**: In contrast to ViTs, large language model pre-training typically involves processing a significantly larger number of training tokens across a vast semantic space (e.g., a vocabulary size of 32,000 in LLaMA-1/2), often without completing a full training epoch. In such case, we empirically find that **Identity Gate** off-the-shelf provides consistent and significant improvements to different low-rank architectures, with zero parameter overhead and only negligible compute overhead (see Section 4.2).
- **Fine-Tuning Task**: The optimization space of fine-tuning is already constrained and therefore very small, where introducing additional parameters is often unnecessary. In these scenarios, we recommend using the **Identity Gate** or **Linear Gate** to preserve training efficiency while enabling information flow across low-rank subspaces.

4 Pre-training Experiments

We first evaluate the performance of LaX in some low-rank pre-training experiments of ViTs/LLMs.

4.1 Pre-training Vision Transformers

We pretrain ViT-Base/Large (224 resolution with 16×16 patch size) and its corresponding low-rank variants on ImageNet-1K [49]. We consider SVD, Tensor Train and CoLA [43] as the baseline low-rank models. All models are trained from scratch for 300 epochs according to the setting of [9]. For SVD and CoLA, we only place Inter-layer LaX with the **Tensor Gate**. For tensor train, we use both Inter-Layer LaX and Intra-Layer LaX with the **Tensor Gate**. More details of the model and training configurations are provided in Appendix A.1.

As shown in Tab. 1, all low-rank baselines experienced accuracy drop compared to the full-rank training. However, LaX consistently improves performance across the three baseline methods (see the training curve in Fig. 6; additional curves in Appendix A.2), with negligible parameter

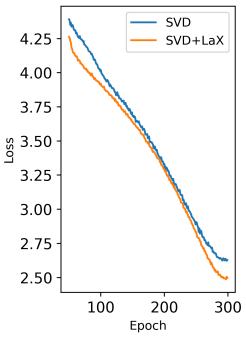

Figure 6: Training Loss

	rank=256		rank=128		rank=64	
Gate	**# Params (M)**	**Accuracy (%)**	**# Params (M)**	**Accuracy (%)**	**# Params (M)**	**Accuracy (%)**
Base Model	44.17	75.20	22.94	74.47	12.32	71.26
+ Identity Gate	44.17	75.81 (+0.61)	22.94	74.65 (+0.18)	12.32	71.64 (+0.38)
+ Linear Gate	44.17	76.11 (+0.91)	22.94	75.31 (+0.84)	12.32	71.72 (+0.46)
+ Tensor Gate	44.24	**77.20** (+2.00)	22.97	**76.35** (+1.88)	12.34	**72.94** (+1.68)
+ Dense Gate	48.55	77.03 (+1.83)	24.04	75.43 (+0.96)	12.60	72.33 (+1.07)

Table 2: Pre-training performance of different LaX Gates on SVD-based ViT-B under varying rank settings. LaX consistently improves performance across all configurations, with the **Tensor Gate** achieving the largest gains while incurring minimal parameter overhead.

overhead ($\leq 0.2\%$). On the ViT-B scale, it recovers the lost performance, boosting accuracy by **+2.00%** to 77.20% for SVD. For Tensor Train, LaX lifts accuracy from 71.11% to 75.43% (**+4.32%**), turning the weakest model into a strong contender. Even for CoLA, the best-performing baseline, LaX adds boosts the accuracy by **1.80%**, reaching 77.84%. Similar trends are observed in ViT-L.

We further evaluate the impact of different LaX Gate variants under varying rank r configurations in ViT pre-training. As shown in Tab. 2, all gate variants consistently improve accuracy across different rank settings compared to their respective base models. Among them, **Tensor Gate** achieves the highest gains with minimal parameter overhead. At rank 256, **Tensor Gate** improves accuracy by +2.00% with only +0.07M additional parameters. As the rank decreases, the added parameter cost also diminishes: at rank $128/64$, the **Tensor Gate** requires only +0.03M/+0.02M more parameters to achieve +1.88%/+1.68% accuracy gains, respectively.

Model	**Computation Complexity**
Original	$\mathcal{O}(nd^2 + n^2d)$
SVD / CoLA	$\mathcal{O}(ndr + n^2d)$
Tensor Train	$\mathcal{O}(ndr + n^2d)$[2]

Table 3: Model Complexity (per block) under batch size 1.

Gate Type	**FLOPs Overhead**
Res / Norm	$\mathcal{O}(nr)$
Identity	$\mathcal{O}(1)$
Linear	$\mathcal{O}(nr)$
Tensor	$\mathcal{O}(nr)$
Dense	$\mathcal{O}(nr^2)$

Table 4: LaX Gate Overhead under batch size 1.

Complexity Analysis We further analyzed the computational complexities (measured by the number of FLOPs in each transformer block) of the original models and the gating mechanisms of LaX. Tab. 3 shows the FLOPs of the models without LaX while Tab. 4 shows the additional FLOPs required by LaX gating, where n is the sequence length, d is the hidden dimension, and r is the rank. As shown by the tables, the computation overhead introduced by *Res*, *Norm*, *Identity*, *Linear*, and *Tensor Gate* is at least an order of magnitude smaller than the original models, and so negligible. *Dense Gate* introduces overhead that is quadratic in r, but it could still be acceptable if $r << d$.

4.2 Pre-training Language Models

We further evaluate LaX in language model pre-training tasks where previous work suggests that pure low-rank architectures often cause performance drop [37, 66, 15]. More recent work such as CoLA [43] and LORO [44], have shown promising results by imposing low-rank activations or performing manifold optimization. Since LORO optimizes $\mathbf{A}$ and $\mathbf{B}$ in the rank-r manifold that $\mathbf{W} = \mathbf{BA}$ lies on, the proposed formulation of LaX contradicts this assumption. Consequently, we compare LaX with SVD and CoLA[3], and directly cite the results reported in [66, 15, 44, 43].

We adopt the same experimental setup from recent benchmarks [66, 15, 44, 43], pre-training LLaMA-like models from 60M to 1B parameters on C4 [48] without data repetition and using compute-optimal token-to-parameter ratios[4]. All linear layers in the original LLaMA architecture are replaced with low-rank layers. For CoLA, we follow [43], and implement the SVD baseline by removing its low-rank activations and/or restoring the original activation. All methods use the same rank for fairness. Full training details are provided in Appendix A.3.

[2]This provides a loose upper bound on complexity, but still tighter than the one reported in [45].

[3]Due to resource constraint, we focus on baseline architectures that performed better in our ViT experiments

[4]The token-to-parameter (T2P) ratios are roughly compute optimal [22].

	60M			130M			350M			1B		
r / d *Tokens*	128 / 512 1.1B			256 / 768 2.2B			256 / 1024 6.4B			512 / 2048 13.1B		
	PPL	Param	Mem	PPL	Param	Mem	PPL	Param	Mem	PPL	Param	Mem
Full-rank	34.06	58	0.43	24.36	134	1.00	18.80	368	2.74	15.56	1339	9.98
ReLoRA [37]	37.04	58	0.37	29.37	134	0.86	29.08	368	1.94	18.33	1339	6.79
GaLore [66]	34.88	58	0.36	25.36	134	0.79	18.95	368	1.90	15.64	1339	6.60
SLTrain [15]	34.15	44	0.32	26.04	97	0.72	19.42	194	1.45	16.14	646	4.81
LORO [44]	33.96	43	0.32	24.59	94	0.70	18.84	185	1.38	15.19	609	4.54
SVD	36.25	43	0.32	26.84	94	0.70	21.18	185	1.38	16.54	609	4.54
SVD + LaX	**33.54** (-2.71)	44	0.33	24.63 (-2.21)	94	0.70	18.90 (-2.28)	185	1.38	15.51 (-1.03)	609	4.54
CoLA [43]	34.04	43	0.32	24.48	94	0.70	19.40	185	1.38	15.52	609	4.54
CoLA + LaX	**33.21** (-0.83)	44	0.33	**24.21** (-0.27)	99	0.74	**18.51** (-0.89)	196	1.46	**14.78** (-0.74)	609	4.54

Table 5: Comparisons of LaX and its base models against other low-rank methods on pre-training C4 dataset [48] from 60M to 1B. We report the validation perplexity (PPL (↓)), number of parameters in millions (Param), and the estimated total memory usage in GB (Mem) excluding activations based on BF16 precision. Results other than LaX and vanilla SVD are from [66, 15, 43, 44].

	LaX		60M		130M		350M	
	Res	Gate	PPL	Rank	PPL	Rank	PPL	Rank
Full-Rank	-	-	34.06	512	24.36	768	18.80	1024
LORO	-	-	33.96	128	24.59	256	18.84	256
SVD – Lower Rank	×	×	36.25	128	26.84	256	21.18	256
	✓	×	33.61 (-2.64)		24.63 (-2.21)		18.90 (-2.28)	
	✓	✓	33.54 (-2.71)		24.66 (-2.18)		18.93 (-2.25)	
CoLA – Lower Rank	×	×	34.04	128	24.48	256	19.40	256
	✓	×	33.82 (-0.22)		24.37 (-0.11)		18.81 (-0.59)	
	✓	✓	**33.21** (-0.83)		**24.21** (-0.27)		**18.51** (-0.89)	
SVD – Higher Rank	×	×	33.45	224	26.20	$\begin{bmatrix}256\\384\end{bmatrix}$	19.68	$\begin{bmatrix}384\\512\end{bmatrix}$
	✓	×	31.62 (-1.83)		23.86 (-2.34)		18.39 (-1.29)	
	✓	✓	31.82 (-1.63)		23.97 (-2.23)		18.21 (-1.47)	
CoLA – Higher Rank	×	×	31.52	224	23.97	$\begin{bmatrix}256\\384\end{bmatrix}$	18.32	$\begin{bmatrix}384\\512\end{bmatrix}$
	✓	×	31.42 (-0.10)		23.74 (-0.23)		17.53 (-0.79)	
	✓	✓	**30.90** (-0.60)		**23.42** (-0.55)		**17.34** (-0.98)	

Table 6: Comparisons of LaX on SVD/CoLA between different Gate variants (×denotes Identity Gate, ✓denotes Dense Gate) and rank choices across 60M to 350M scales. For scenarios where a vector of ranks is provided, smaller one is for attention layers and the larger one is for MLP layers.

As shown in Tab. 5, LaX improves the validation perplexity of SVD and CoLA across all scales. In particular, vanilla SVD performs poorly compared to most baselines but can be boosted to perform on par with or surpassing LORO and CoLA. While CoLA perform similarly to LORO, its LaX -boosted version surpasses LORO on all scales. In addition, LaX just uses the standard Adam optimizer and does not need LORO's complex manifold gradient computations and deeply customized training strategies[5].

Tab. 6 compares SVD/CoLA variants across different ranks and LaX configurations. From Tab. 6 we observe that with either Identity Gate (denoted by ×) or Dense Gate (denoted by ✓), LaX continues to increase performance. In particular, the results in Tab. 6 further demonstrate that LaX is consistently effective regardless of whether the base model has a higher rank. The trend continues to hold that LaX boosts more on a weaker base model than on a stronger base model.

The only mixed message in Tab. 6 is the effectiveness of LaX Gate. In CoLA from 60M to 350M, the Dense Gate consistently outperforms the Identity Gate. However, for the SVD method, a Dense Gate does not provide a consistent benefit. When results are mixed, the rule-of-thumb is to be conservative; therefore, we recommend using **Identity Gate** in language model pre-training, as it

[5]LORO requires periodical computations of manifold gradient which involves tuning the update frequency, at each update step a learning rate warm-up and a refreshment of Adam statistics.

Model	Method ($r = 32$)	# Params (%)	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	Avg
LLaMA-7B	LoRA	0.83	95.0	36.1	84.3	17.7	84.4	51.8	61.6
	LoRA + LaX (Ours)	0.83	**96.9** (+1.9)	**37.7** (+1.6)	**84.8** (+0.5)	**19.3** (+1.6)	**87.8** (+3.4)	**53.6** (+1.8)	**63.4** (+1.8)
LLaMA-13B	LoRA	0.67	95.2	47.5	86.0	18.2	89.8	54.6	65.2
	LoRA + LaX (Ours)	0.67	**97.3** (+2.1)	**49.0** (+1.5)	**86.3** (+0.3)	**20.9** (+2.7)	**91.9** (+2.1)	**58.3** (+3.7)	**67.3** (+2.1)

Table 7: Accuracy comparison of LoRA and LaX-LoRA on six math reasoning datasets.

Model	Method ($r = 32$)	# Params (%)	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg
LLaMA-7B	LoRA	0.83	68.9	80.7	77.4	78.1	78.8	77.8	61.3	74.8	74.7
	LoRA + LaX (Ours)	0.83	**69.6** (+0.7)	**81.9** (+1.2)	**78.9** (+1.5)	**84.7** (+6.6)	**80.8** (+2.0)	**79.8** (+2.0)	**64.8** (+3.5)	**78.0** (+3.2)	**77.3** (+2.6)
LLaMA-13B	LoRA	0.67	**72.1**	83.5	80.5	90.5	83.7	82.8	68.3	82.4	80.5
	LoRA + LaX (Ours)	0.67	71.3 (-0.8)	**85.4** (+1.9)	**81.3** (+0.8)	**91.3** (+0.8)	**84.1** (+0.4)	**84.4** (+1.6)	**71.8** (+3.5)	**83.1** (+0.7)	**81.6** (+1.1)

Table 8: Comparison of LoRA and LaX-LoRA on Commonsense Reasoning Benchmarks.

already effectively boosts SVD and CoLA architectures. Consequently, we conduct the pre-training experiments on the 1B scale using the Identity Gate.

On larger-scale CoLA (e.g., 350M and 1B), LaX tends to bring more benefit, contrary to the stable or decreasing trend observed in SVD. This may be caused by the architectural difference between SVD and CoLA, indicating that CoLA might benefit more from LaX when it scales up.

5 Fine-tuning Experiments

Finally we show the effectiveness of LaX in low-rank fine-tuning. We incorporate LaX into LoRA (Fig. 2 (d)) and consider two widely used reasoning benchmarks [41, 25]: Arithmetic/Commonsense Reasoning. The fine-tuning configuration in this section strictly follows [24]. Additional configuration details are provided in Appendix A.4.

5.1 Arithmetic Reasoning

We fine-tune LLaMA-7B /13B on the Math10K dataset and assess performance in six arithmetic reasoning subtasks. For this evaluation, we configure LaX with Linear Gate.

Tab. 7 shows that augmenting LoRA with LaX yields consistent accuracy improvements across all six arithmetic subtasks for both LLaMA-7B and LLaMA-13B. For the 7B model, the average score improves from 61.6% to 63.4% (+1.8%), while for the 13B model it increases from 65.2% to 67.3% (+2.1%), suggesting that LaX's rank expansion mechanism effectively provides additional representation capacity required by fine-tuning. Even on subtasks where LoRA already performs strongly (e.g. MultiArith, AddSub), LaX delivers consistent improvements.

5.2 Commonsense Reasoning

Following [25, 41], we merge the training datasets from all eight commonsense reasoning tasks into a unified training set and evaluate the performance separately on each task. In this experiment, we configure LaX with the Identity Gate. As shown in Tab. 8, LaX consistently outperforms the LoRA baseline in all reasoning tasks. For LLaMA-7B, the average accuracy increases from 74.7% to 77.3% (+2.6%), with a notable gain of +6.6% on HellaSwag. For LLaMA-13B, the overall score rises by +1.1%; only BoolQ exhibits a marginal decline (-0.8%).

6 Conclusion

In this work, we have presented **Latent Crossing** (LaX), a lightweight and versatile module designed to improve the training performance of low-rank compressed models. Although low-rank methods are effective in reducing computational overhead, they often suffer from a loss in model expressiveness and performance. LaX addresses this limitation by enabling information flow between low-rank subspaces through residual connections equipped with simple gating mechanisms. As a result, LaX serves as a general plug-in booster that enhances a wide range of low-rank models, across both pretraining and fine-tuning scenarios, and for both language and vision tasks.

References

[1] L. Balzano, T. Ding, B. D. Haeffele, S. M. Kwon, Q. Qu, P. Wang, Z. Wang, and C. Yaras. An overview of low-rank structures in the training and adaptation of large models. *arXiv preprint arXiv:2503.19859*, 2025.

[2] J. Biamonte and V. Bergholm. Tensor networks in a nutshell. *arXiv preprint arXiv:1708.00006*, 2017.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. *Psychometrika*, 35(3):283–319, 1970.

[5] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[6] A. Cichocki. Era of big data processing: A new approach via tensor networks and tensor decompositions. *arXiv preprint arXiv:1403.2048*, 2014.

[7] M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A. P. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pages 7480–7512. PMLR, 2023.

[8] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[10] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[11] R. Feng, K. Zheng, Y. Huang, D. Zhao, M. Jordan, and Z.-J. Zha. Rank diminishing in deep neural networks. *Advances in Neural Information Processing Systems*, 35:33054–33065, 2022.

[12] T. Garipov, D. Podoprikhin, A. Novikov, and D. P. Vetrov. Ultimate tensorization: compressing convolutional and fc layers alike. *CoRR*, abs/1611.03214, 2016.

[13] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[14] A. Graves and A. Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[15] A. Han, J. Li, W. Huang, M. Hong, A. Takeda, P. Jawanpuria, and B. Mishra. SLTrain: a sparse plus low rank approach for parameter and memory efficient pretraining. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[16] R. A. Harshman et al. Foundations of the parafac procedure: Models and conditions for an explanatory multi-modal factor analysis. *UCLA working papers in phonetics*, 16(1):84, 1970.

[17] S. Hayou, N. Ghosh, and B. Yu. LoRA+: Efficient low rank adaptation of large models. In *Forty-first International Conference on Machine Learning*, 2024.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer, 2016.

[20] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 558–567, 2019.

[21] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[22] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[23] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[24] Z. Hu, L. Wang, Y. Lan, W. Xu, E.-P. Lim, L. Bing, X. Xu, S. Poria, and R. Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5276, Singapore, Dec. 2023. Association for Computational Linguistics.

[25] Z. Hu, L. Wang, Y. Lan, W. Xu, E.-P. Lim, L. Bing, X. Xu, S. Poria, and R. K.-W. Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[26] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[27] S. R. Kamalakara, A. Locatelli, B. Venkitesh, J. Ba, Y. Gal, and A. N. Gomez. Exploring low rank training of deep neural networks. *CoRR*, abs/2209.13569, 2022.

[28] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[29] H. A. Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 14(3):105–122, 2000.

[30] H. Kim, M. U. K. Khan, and C.-M. Kyung. Efficient neural network compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12569–12577, 2019.

[31] Y. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. In Y. Bengio and Y. LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[32] D. Kressner and C. Tobler. htuckera matlab toolbox for tensors in hierarchical tucker format. *Mathicse, EPF Lausanne*, page 11, 2012.

[33] T. Kumar, Z. Ankner, B. F. Spector, B. Bordelon, N. Muennighoff, M. Paul, C. Pehlevan, C. Re, and A. Raghunathan. Scaling laws for precision. In *The Thirteenth International Conference on Learning Representations*, 2025.

[34] V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, and V. S. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. 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.

[35] J. Li and V. Papyan. Residual alignment: uncovering the mechanisms of residual networks. *Advances in Neural Information Processing Systems*, 36:57660–57712, 2023.

[36] S. Li, P. Zhang, G. Gan, X. Lv, B. Wang, J. Wei, and X. Jiang. Hypoformer: Hybrid decomposition transformer for edge-friendly neural machine translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7056–7068, 2022.

[37] V. Lialin, S. Muckatira, N. Shivagunde, and A. Rumshisky. ReloRA: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations*, 2024.

[38] V. Lialin, N. Shivagunde, S. Muckatira, and A. Rumshisky. Relora: High-rank training through low-rank updates. *arXiv preprint arXiv:2307.05695*, 2023.

[39] L. Liebenwein, A. Maalouf, D. Feldman, and D. Rus. Compressing neural networks: Towards determining the optimal layer-wise decomposition. *Advances in Neural Information Processing Systems*, 34:5328–5344, 2021.

[40] P. Liu, Z.-F. Gao, W. X. Zhao, Z.-Y. Xie, Z.-Y. Lu, and J.-R. Wen. Enabling lightweight fine-tuning for pre-trained language model compression based on matrix product operators. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *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 5388–5398, Online, Aug. 2021. Association for Computational Linguistics.

[41] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024.

[42] X. Liu, J. Su, and F. Huang. Tuformer: Data-driven design of transformers for improved generalization or efficiency. In *The Tenth International Conference on Learning Representations (ICLR 2022)*, 2022.

[43] Z. Liu, R. Zhang, Z. Wang, Z. Yang, P. Hovland, B. Nicolae, F. Cappello, and Z. Zhang. Cola: Compute-efficient pre-training of llms via low-rank activation. *arXiv preprint arXiv:2502.10940*, 2025.

[44] Z. Mo, L.-K. Huang, and S. J. Pan. Parameter and memory efficient pretraining via low-rank riemannian optimization. In *The Thirteenth International Conference on Learning Representations*.

[45] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov. Tensorizing neural networks. In *Advances in Neural Information Processing Systems 28 (NIPS)*. 2015.

[46] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[47] X. Ou, Z. Chen, C. Zhu, and Y. Liu. Low rank optimization for efficient deep learning: Making a balance between compact architecture and fast training. *Journal of Systems Engineering and Electronics*, 35(3):509–531, 2023.

[48] 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.

[49] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[50] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac. Computational complexity of projected entangled pair states. *Physical review letters*, 98(14):140506, 2007.

[51] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[52] M. A. O. Vasilescu. Causal deep learning: Causal capsules and tensor transformers. *CoRR*, abs/2301.00314, 2023.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[54] F. Verstraete, V. Murg, and J. I. Cirac. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in physics*, 57(2):143–224, 2008.

[55] T. Wang, Z. Dou, C. Bao, and Z. Shi. Diffusion mechanism in residual neural network: theory and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(2):667–680, 2023.

[56] W. Wang, Y. Sun, B. Eriksson, W. Wang, and V. Aggarwal. Wide compression: Tensor ring nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9329–9338, 2018.

[57] B. Wu, D. Wang, G. Zhao, L. Deng, and G. Li. Hybrid tensor decomposition in neural network compression. *Neural Networks*, 132:309–320, 2020.

[58] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[59] Y. Yang, J. Zhou, N. Wong, and Z. Zhang. LoRETTA: Low-rank economic tensor-train adaptation for ultra-low-parameter fine-tuning of large language models. In K. Duh, H. Gomez, and S. Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3161–3176, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[60] Z. Yang, Z. Liu, S. Choudhary, X. Xie, C. Gao, S. Kunzmann, and Z. Zhang. Comera: Computing-and memory-efficient training via rank-adaptive tensor optimization. *Advances in Neural Information Processing Systems*, 37:77200–77225, 2024.

[61] X. Yu, T. Liu, X. Wang, and D. Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7370–7379, 2017.

[62] L. Zhang, L. Zhang, S. Shi, X. Chu, and B. Li. LoRA-FA: Memory-efficient low-rank adaptation for large language models fine-tuning, 2024.

[63] Q. Zhang, M. Chen, A. Bukharin, P. He, Y. Cheng, W. Chen, and T. Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023.

[64] Q. Zhang, R. Zhang, J. Sun, and Y. Liu. How sparse can we prune a deep network: A fundamental limit perspective. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 91337–91372. Curran Associates, Inc., 2024.

[65] X. Zhang, J. Zou, K. He, and J. Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):1943–1955, 2016.

[66] J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, and Y. Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

[67] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, 2016.

A Hyperparameter

A.1 ViTs Pre-training Configurations

Models	LaX	Rank	Epochs	Base LR	LR decay	Weight decay	Dropout	Warmup
ViT-B	–	–	300	3e-3	cosine	0.3	0.1	10
SVD	× ✓	256	300	1e-3	cosine	0.3	0.1	10
TT	× ✓	336						
CoLA	× ✓	256						

Table 9: Training hyperparameter for ViT-B and its low-rank variants on ImageNet-1k.

Models	LaX	Rank	Epochs	Base LR	LR decay	Weight decay	Dropout	Warmup
ViT-L	–	–	300	3e-3	cosine	0.3	0.1	10
SVD	× ✓	256	300	1e-3	cosine	0.3	0.1	10
TT	× ✓	256						
CoLA	× ✓	256						

Table 10: Training hyperparameter for ViT-L and its low-rank variants on ImageNet-1k.

Models	Inter-Layer LaX	Intra-Layer LaX
SVD	Tensor Gate	×
TT		Tensor Gate
CoLA		×

Table 11: LaX gating variants for different low-rank methods.

Models	Inter-Layer LaX	Intra-Layer LaX
SVD		$\times$
TT	QKV+MLP	QKV+MLP
CoLA		$\times$

Table 12: LaX placed layers for different low-rank methods.

Models	# Cores	QKV d_i	MLP1 d_i	MLP2 d_i
ViT-B	4	{32,24,24,32}	{32,24,48,64}	{48,64,24,32}
ViT-L	4	{32,32,32,32}	{32,32,64,64}	{64,64,32,32}

Table 13: Tensor Train Dimension Configuration for ViTs.

A.2 ViT Pre-training Loss Curves

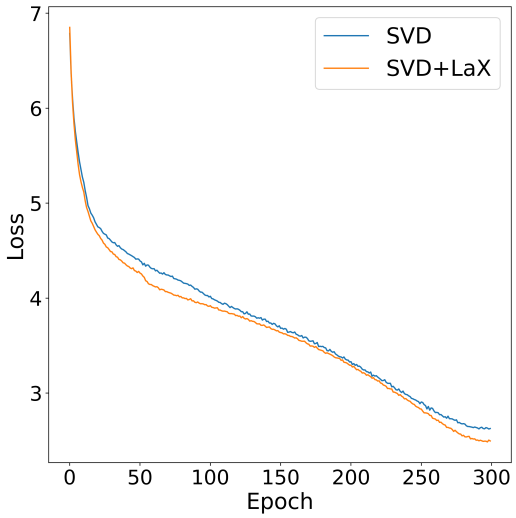

Figure 7: Pre-training Loss Curve of SVD and SVD+LaX on ImageNet-1k

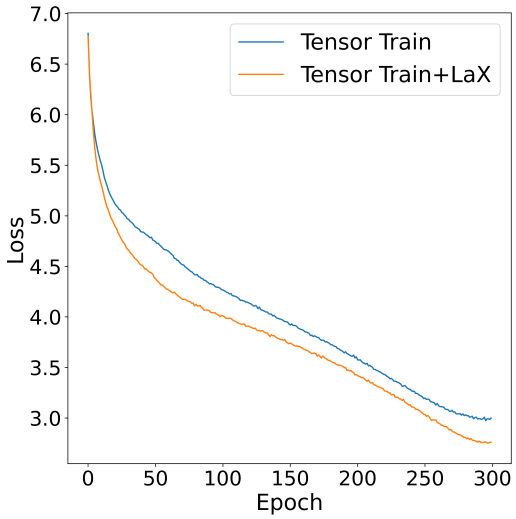

Figure 8: Pre-training Loss Curve of Tensor Train and Tensor Train+LaX on ImageNet-1k

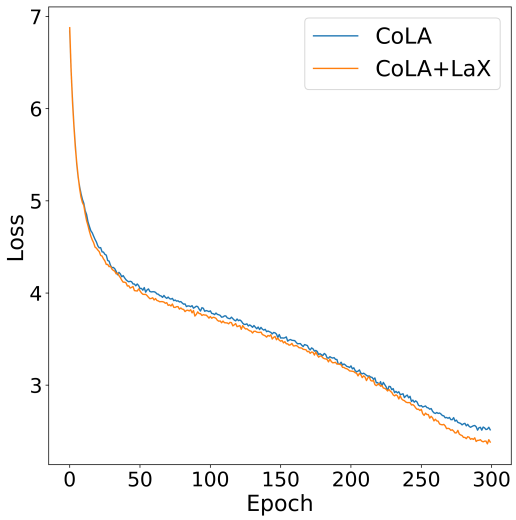

Figure 9: Pre-training Loss Curve of CoLA and CoLA+LaX on ImageNet-1k

A.3 LLMs Pre-training Configurations

Scales	Models	LaX	Steps	Base LR	LR decay	Weight decay	Warmup	Gradient clipping
60M	SVD	×	10k	2e-3	cosine	0.01	2k	0.5
		✓		2e-2				
	CoLA	×		6e-3				
		✓		4e-2				
130M	SVD	×	20k	1e-3	cosine	0.01	4k	0.5
		✓		1e-2				
	CoLA	×		4e-3				
		✓		2e-2				
350M	SVD	×	60k	1e-3	cosine	0.01	12k	0.5
		✓		1e-2				
	CoLA	×		3e-3				
		✓		2e-2				
1B	SVD	×	100k	8e-4	cosine	0.01	20k	0.5
		✓		3e-3				
	CoLA	×		2e-3				
		✓		1e-2				

Table 14: Hyper-parameters for pre-training SVD and CoLA and their LaX variants for LLaMA-like models from 60M to 1B.

The primal hyper-parameter for LLM pre-training experiments is the learning rate. For small models such as 60M and 130M, we typically sweep at the scale of 1e-3 for base models and 1e-2 for their LaX variants. The rule-of-thumb that we empirically found is to choose the largest learning rate that does not cause divergence issues. In particular, SVD base models are severely more sensitive to learning rate, and can only afford smaller settings compared to base CoLA. For both SVD and CoLA, LaX offers additional stability that facilitates an order of magnitude larger learning rates. The experience of training smaller models are then adopted for larger scales such as 350M and 1B, which continue following the trend that the proper choice of learning rate decreases when model scale increases. For the same scale, we did not find evident that further tuning learning rates are beneficial. Consequently, we adopt the same setting when only changing the rank for each scale.

A.4 LaX-LoRA Fine-tuning Configurations

Hyperparameters (LoRA)	LLaMA-7B	LLaMA-13B
Rank r	32	32
α	64	64
Dropout	0.0	0.0
Optimizer	AdamW	AdamW
LR	3e-4	3e-4
Scheduler	Linear	Linear
Batch size	16	16
Accumulation steps	4	4
Cut off length	256	256
Warmup steps	100	100
Epochs	3	3
Where	Q,K,V,Up,Down	Q,K,V,Up,Down

Table 15: Commonsense Hyperparameter settings for LoRA on LLaMA-7B and LLaMA-13B.

Hyperparameters (LaX-LoRA)	LLaMA-7B	LLaMA-13B
Rank r	32	32
α	64	64
Dropout	0.0	0.0
Optimizer	AdamW	AdamW
LR	3e-4	3e-4
Scheduler	Linear	Linear
Batch size	16	16
Accumulation steps	4	4
Cut off length	256	256
Warmup steps	100	100
Epochs	3	3
Where	Q,K,V,Up,Down	Q,K,V,Up,Down
Where LaX	Q,K,V,Up,Down	Q,K,V,Up,Down
LaX Gate	Identity	Identity

Table 16: Commonsense Hyperparameter settings for LaX-LoRA on LLaMA-7B and LLaMA-13B.

Hyperparameters (LoRA)	LLaMA-7B	LLaMA-13B
Rank r	32	32
α	64	64
Dropout	0.0	0.0
Optimizer	AdamW	AdamW
LR	3e-4	3e-4
Scheduler	Linear	Linear
Batch size	16	16
Accumulation steps	4	4
Cut off length	256	256
Warmup steps	100	100
Epochs	3	3
Where	Q,K,V,Up,Down	Q,K,V,Up,Down

Table 17: Arithmetic Hyperparameter settings for LoRA on LLaMA-7B and LLaMA-13B.

Hyperparameters (LaX-LoRA)	LLaMA-7B	LLaMA-13B
Rank r	32	32
α	64	64
Dropout	0.0	0.0
Optimizer	AdamW	AdamW
LR	3e-4	3e-4
Scheduler	Linear	Linear
Batch size	16	16
Accumulation steps	4	4
Cut off length	256	256
Warmup steps	100	100
Epochs	3	3
Where	Q,K,V,Up,Down	Q,K,V,Up,Down
Where LaX	Q,K,V,Up,Down	Q,K,V,Up,Down
LaX Gate	Linear	Linear

Table 18: Arithmetic Hyperparameter settings for LaX-LoRA on LLaMA-7B and LLaMA-13B.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Section 3.1. Empirical validation is in Section 4.1, and 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 4.2. We discussed the limitation regarding gating module in LLMs pre-training tasks.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: This paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We release our code link in this paper. All the datasets used are public, and we provide full hyperparameters in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have provided open access to our code, the anonymized URL is included. All the datasets used are public, and we provide full hyperparameters in the Appendix.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so No is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 4.1 and Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Pre-training large foundation models is highly computationally expensive. For instance, a single pre-training run of an SVD-based ViT-B model under our setup requires approximately 450 A100 GPU hours. Larger models used in this study incur even higher costs. Due to computational constraints, we are unable to conduct multiple pre-training runs for all variants.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Section 5. We provide the number of parameters for all experimented models, and we also provide the run memory in Tab.5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We confirm that our research adheres to the NeurIPS Code of Ethics in every respect and preserves anonymity.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Our paper proposes new methods for efficient training of large models. The research has the possibility to significantly reduce the training costs of large models and also accelerate the inference.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our research didnt release any data or models at such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes, they are properly credited, mentioned, and respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Our work introduces no new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We confirm that the proposed method in our paper does not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.