SINGLORA: LOW RANK ADAPTATION USING A SINGLE MATRIX

David Bensaid[*1], **Noam Rotstein**[1], **Roy Velich**[2], **Daniel Bensaid**[3], **Ron Kimmel**[1,2]
[1]Department of Electrical Engineering, Technion - Israel Institute of Technology
[2]Department of Computer Science, Technion - Israel Institute of Technology
[3]Paris Dauphine

ABSTRACT

Low-Rank Adaptation (LoRA) has significantly advanced parameter-efficient fine-tuning of large pretrained models. LoRA augments the pre-trained weights of a model by adding the product of two smaller matrices that together form a low-rank matrix update. Recent research has shown that scale disparities between these two matrices often cause unstable training dynamics, leading to suboptimal performance. In this paper, we reformulate low-rank adaptation by learning the weights update as a decomposition of a **single** low-rank matrix multiplied by its transpose. This design, called SINGLORA, inherently removes inter-matrix scale conflicts, ensuring stable optimization, and roughly halves the parameter count. We analyze SINGLORAwithin the efficient feature learning framework, showing that it guarantees stability by construction while preserving the expressive capacity of LoRA in the transformer architecture. Extensive experiments on multiple tasks validate these benefits. In common sense reasoning, fine-tuning LLama 7B on MNLI with SINGLORAachieves 91.3% accuracy — surpassing LoRA (89.1%) and LoRA+ (90.2%) — while using only 60% of their parameter budget. In image generation, fine-tuning Stable Diffusion with SINGLORAsignificantly improves image fidelity on DreamBooth, achieving a DINO similarity score of 0.151, compared to scores of 0.148 and 0.143 for DoRA and LoRA, respectively.

1 INTRODUCTION

Adapting large pretrained models for specialized tasks has emerged as a central focus in machine learning research. These efforts aim to take advantage of the strong generalization capabilities of such models while meeting domain-specific requirements. Rapid scaling of model sizes and datasets has made full fine-tuning computationally prohibitive, driving the development of parameter-efficient fine-tuning (PEFT) methods to reduce computational costs. Among PEFT approaches, Low-Rank Adaptation (LoRA) (6) has gained particular popularity due to its simplicity and effectiveness in various domains. LoRA augments pre-trained weights matrices $\mathbf{W}_0 \in \mathbb{R}^{d\times k}$ with low-rank updates,

$$\mathbf{W}_0 + \frac{\alpha}{r}\mathbf{B}\mathbf{A},$$

where $\mathbf{B} \in \mathbb{R}^{d\times r},\ \mathbf{A} \in \mathbb{R}^{r\times k},\ \alpha \in \mathbb{R}$ and $r \ll \min(d, k)$ is the rank of $\mathbf{A}$ and $\mathbf{B}$.

Recent research (4; 21; 22) has identified scale disparities between matrices $\mathbf{A}$ and $\mathbf{B}$ as a fundamental limitation of LoRA. Specifically, the norms of $\mathbf{A}$ and $\mathbf{B}$ often differ significantly in magnitude during training, creating imbalanced gradient flows that lead to unstable training dynamics. Such scale imbalances cause vanishing or exploding gradients and ultimately result in suboptimal performance. While recent works have identified this challenge, existing solutions require careful tuning of different learning rates for $\mathbf{A}$ and $\mathbf{B}$ (4) or the use of specialized optimizers (21; 22), making them challenging to adopt in practice.

*Corresponding author: dben-said@campus.technion.ac.il

In this paper, we introduce SINGLORA, which reformulates the low-rank adaptation paradigm using a *single* low-rank matrix $A \in \mathbb{R}^{n \times r}$ that produces the symmetric update,

$$\mathbf{W}_0 + \frac{\alpha}{r}\mathbf{A}\mathbf{A}^\top. \tag{1}$$

This formulation provides key benefits over LoRA and its recent variants. First, it ensures stable optimization by design, eliminating inter-matrix scale conflicts without requiring extensive hyperparameter tuning or custom optimizers. Second, it achieves a parameter reduction by roughly halving the number of learned parameters while demonstrating better performance.

A theoretical analysis of SINGLORAin the efficient feature learning setting (3; 4) demonstrates that, unlike LoRA, it guarantees *stable feature learning*. We then show that the symmetric parameterization of SINGLORA, which might initially appear to limit expressiveness, preserves the full expressive capacity of LoRA in the transformer architecture. Building on this theoretical foundation, we extend SINGLORAto non-square matrices, ensuring its applicability to any neural network and validate its results through comprehensive experiments across multiple modalities. In comprehension reasoning, fine-tuning LLaMA (16) on MNLI (18) with SINGLORAoutperforms LoRA and LoRA+ (91.3% against 89.1% and 90.2% respectively) while using only 60% of their parameter budget. In image generation, SINGLORAincreases the image fidelity of a Stable Diffusion model finetuned on Dreambooth (14) by 5.4% compared to DoRA (9), the current state-of-the-art low rank adapter.

2 RELATED EFFORTS

2.1 LORA AND ITS EXTENSIONS

As large multimodal models continue to scale, an increasing number of parameter-efficient fine-tuning (PEFT) techniques have been developed to facilitate their adaptation to downstream tasks. LoRA has emerged as one of the most popular adapter, offering efficient model adaptation across various domains. It has been studied extensively, resulting in numerous variations. DyLoRA (17) and AdaLoRA (23) focus on rank adjustment during training, with adaptive strategies to optimize rank allocation. LoHA (7) and LoKR (2), propose architectural extensions and, respectively leverage Hadamard and Kronecker products of two rank-r approximations to obtain more expressive adapters. Delta-LoRA (25) modifies LoRA by updating pre-trained weights using differences in successive low-rank updates, and LoRA-Drop (24) selects the most impactful LoRA adapters to reduce computational cost. More recently, Weight-Decomposed Low-Rank Adaptation (DoRA) (9) sets state-of-the-art performance by decomposing the pre-trained weight matrix into magnitude and direction components before employing LoRA to update the directional component. Our approach is complementary to these extensions and can be seamlessly integrated with DoRA and other variants to further enhance their effectiveness by resolving the underlying optimization instabilities.

2.2 LORA'S STABILITY ISSUE

A recent line of research has identified fundamental limitations in standard optimizers, such as Stochastic Gradient Descent (SGD) (12) and Adam (8), when applied to LoRA modules. In training, the norms of $\mathbf{A}$ and $\mathbf{B}$ often differ significantly in magnitude, leading to unstable training dynamics. LoRA+ (4) shows that $\mathbf{A}$ and $\mathbf{B}$ should be optimized with different learning rates to ensure stable learning dynamics. Building on this theoretical foundation, Zhang *et al.* (22) have proposed to use Riemannian gradient descent and conditioning methods to stabilize the optimization process. Recently, LoRADoneRite (21) identified the multiplicity of possible optimizer updates for a single low-rank adapter as the source of instability in LoRA's training. Building on these theoretical insights, we analyze the convergence properties of SINGLORA. We show that its streamlined low-rank adaptation paradigm naturally promotes stable training dynamics, without requiring careful learning rate tuning (4) or modifications to classical optimizers (21).

3 SINGLORA

In this section, we formally present the core formulation of SINGLORA. We then analyze its training dynamics proving that, unlike LoRA and its recent variants, it guarantees stable feature learning. We establish formal convergence properties under standard optimizers such as SGD (12), Adam (8) and AdamW (10). Finally, SINGLORAis extended to non-square weight matrices, making it applicable across diverse neural architectures.

3.1 SINGLORA'S FORMULATION

SINGLORAreformulates low-rank adaptation by replacing the traditional two-matrix decomposition with a single learnable matrix. Given a pre-trained model with frozen weight matrix $W_0 \in \mathbb{R}^{n\times n}$, SINGLORAcomputes the adapted weights as,

$$\mathbf{W_0} + \frac{\alpha}{r} u(t)\mathbf{A}\mathbf{A}^\top, \tag{2}$$

where $\mathbf{A} \in \mathbb{R}^{n\times r}$ is a low-rank trainable matrix with rank $r \ll n$, α is a scaling factor, and $u(t)$ is a scalar function controlling the adaptation rate at optimization step t. This formulation provides two key advantages: **(1)** it eliminates inter-matrix scale conflicts by construction, ensuring stable optimization, and **(2)** it reduces the parameter count by roughly half compared to standard LoRA.

Initialization scheme. To enable effective gradient flow during training, we initialize A with a Kaiming distribution (5). To preserve the behavior of the pretrained model at initialization, we require $u(0) = 0$, ensuring that the model starts from the pretrained weights. In practice, we adopt a simple ramp function for $u(t)$, namely $u(t) = \min\left(\frac{t}{T}, 1\right)$, where $T \in \mathbb{R}$ controls the adaptation rate. u provides a smooth transition from the pretrained weights to the adapted model, allowing for gradual incorporation of task-specific features while maintaining stability during the early training stages.

Dataset	Method	RoBERTa			GPT-2		
		Learning rate	Acc. (%)	Params	Learning rate	Acc. (%)	Params
QQP	LoRA	$2e^{-4}$	88.5	0.15M	$4e^{-4}$	87.9	1.78M
	LoRA+	$(2e^{-4}, 4e^{-3})$	89.1	0.15M	$(2e^{-4}, 4e^{-3})$	89.1	1.78M
	LoRADoneRite	$1e^{-3}$	87.4	0.15M	$1e^{-3}$	86.6	1.78M
	DoRA	$5e^{-4}$	89.2	0.16M	$5e^{-4}$	89.2	1.78M
	Ours	$1e^{-3}$	88.9	0.075M	$1e^{-3}$	88.8	0.89M
QNLI	LoRA	$4e^{-4}$	90.9	0.15M	–	–	–
	LoRA+	$(2e^{-4}, 4e^{-3})$	92.1	0.15M	–	–	–
	LoRADoneRite	$1e^{-3}$	91.2	0.15M	–	–	–
	DoRA	$5e^{-4}$	92.1	0.16M	–	–	–
	Ours	$1e^{-3}$	92.2	0.075M	–	–	–
MNLI	LoRA	$4e^{-4}$	85.6	0.15M	$2e^{-4}$	81.3	1.78M
	LoRA+	$(5e^{-5}, 4e^{-3})$	86.5	0.15M	$(2e^{-4}, 4e^{-3})$	82.0	1.78M
	LoRADoneRite	$1e^{-3}$	83.7	0.15M	$1e^{-3}$	80.6	1.78M
	DoRA	$5e^{-4}$	86.4	0.16M	$5e^{-4}$	82.2	1.78M
	Ours	$1e^{-3}$	86.5	0.075M	$1e^{-3}$	82.5	0.89M
Mean	LoRA	–	88.3	0.15M	–	84.6	1.78M
	LoRA+	–	**89.2**	0.15M	–	85.6	1.78M
	LoRADoneRite	–	87.4	0.16M	–	83.6	1.78M
	DoRA	–	**89.2**	0.16M	–	**85.7**	1.78M
	Ours	–	**89.2**	**0.075M**	–	**85.7**	**0.89M**

Table 1: Accuracy of RoBERTa and GPT-2 fine-tuned on GLUE datasets with rank 8 updates. LoRA+ uses learning rates (μ_A, μ_B). GPT-2 results on QNLI were not reported in (4).

3.2 TRANSFORMATION INVARIANCE OF LOW RANK ADAPTERS

In this subsection, we examine the stability properties of SINGLORAin the known infinite-width configuration (15; 19; 20; 3), where the network width n tends to the infinite. We investigate the changes in the model output, denoted Δf, between training iterations. A training process exhibit *stable features learning* if Δf remain bounded and non-vanishing as the network width grows, e.g. $\Delta f = \Theta_n(1)$. We refer the reader to (4; 3) for a complete discussion of the theoretical foundations of the stable feature learning theory.

Observing that pairs of matrices (A_1, B_1) and (A_2, B_2) representing the same adapter, i.e. $A_1B_1 = A_2B_2$, can produce different optimizer updates, Yen *et al.* (21) recently introduced the notion of *transformation-invariance* for low rank adapters.

Definition 1. Transformation-Invariance Let (A_1, B_1) and (A_2, B_2) be LoRA adapters satisfying $A_1B_1 = A_2B_2$. An optimizer is *transformation-invariant* if its updates $(\delta A_1, \delta B_1)$ and $(\delta A_2, \delta B_2)$ satisfy,

$$(A_1 + \delta A_1)(B_1 + \delta B_1) = (A_2 + \delta A_2)(B_2 + \delta B_2) \tag{3}$$

(21) show that the following conditions are sufficient for transformation invariance,

$$\begin{array}{lrcl} \textbf{(i)} & \delta A_1 B_1 & = & \delta A_2 B_2 \\ \textbf{(ii)} & A_1 \delta B_1 & = & A_2 \delta B_2 \\ \textbf{(iii)} & \delta A_1 \delta B_1 & = & \delta A_2 \delta B_2 \end{array} \tag{4}$$

To illustrate how the multiplicity in LoRA's parametrization can yield different gradient descent updates, we consider two parameterizations (A_1, B_1) and (A_2, B_2) of a low-rank adapter related by a scaling factor $s \in \mathbb{R}$. Namely,

$$A_2 = s\, A_1, \quad B_2 = \frac{1}{s}\, B_1. \tag{5}$$

Defining $Z = A_1B_1 = A_2B_2$ and applying the chain rule yields $\nabla A_1 = \nabla Z B_1$, $\nabla A_2 = \nabla Z B_2$, where ∇P stands for the gradient of the loss $\mathcal{L}$ of the model with respect to the parameter P. We can thus rewrite, $\nabla A_2 = \frac{1}{s}\nabla Z B_1 = \frac{1}{s}\nabla A_1$. Therefore,

$$\delta A_1 B_1 = -\eta\, \nabla A_1 B_1 = -\eta\, s\, \nabla A_2 B_1 = s^2\, \delta A_2 B_2,$$

making Eq. 3 unsatisfied in the general case. This example reveals why LoRA exhibits unstable training dynamics when using optimizers that are not transformation-invariant: when the scaling factor s is much larger than 1, the matrices A and B (and their corresponding updates) operate at vastly different scales. This scale mismatch creates a fundamental problem: first-order optimizers using a single learning rate struggle to achieve stable feature learning, as they cannot simultaneously accommodate both large and small-scale updates effectively. This issue frequently arises during training since LoRA's matrices A and B are typically initialized with different scales.

Theorem 1. Any transformation invariant optimizer applying the same update rule for A and B achieves efficient feature learning. *A proof is presented in the appendix.*

Recent efforts (21; 22) attempt to address the stability issues of LoRA by building a dedicated scale-invariant optimizer. In contrast, SINGLORAformulation inherently solves those challenges and SINGLORAcan be efficiently tuned with standard deep learning optimizers, such as SGD or Adam (8), without requiring special modifications or careful hyper-parameters tuning.

Theorem 2. A gradient descent optimizer is transformation-invariant for SINGLORA . *A proof is presented in the appendix.*

Theorem 1 and Theorem 2 guarantee the existence of a learning rate yielding stable dynamics when training SINGLORAwith first-order optimization methods.

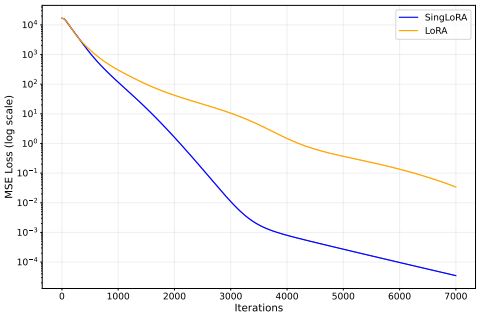

Figure 1: Synthetic experiment: convergence plot for LoRA and SINGLORA.

3.3 EXTENSION TO NON-SQUARE MATRICES

Our discussion has so far focused on square weight matrices $W_0 \in \mathbb{R}^{n\times n}$ which are commonly used in the attention layers of transformer architecture. We now extend our approach to rectangular weight matrices $W_0 \in \mathbb{R}^{d_{\text{in}}\times d_{\text{out}}}$. Without loss of generality, we assume $d_{\text{in}} < d_{\text{out}}$.

Considering a low rank matrix $A \in \mathbb{R}^{d_{\text{out}}\times r}$ we define $A^* \in \mathbb{R}^{d_{\text{in}}\times r}$ as a truncation of A consisting of its first d_{in} rows. The adapted layer is then computed by $W_0 + A^*A^\top$. Training this adapter with standard optimizers preserves the stability and transformation-invariance properties demonstrated for the square case.

Theorem 3. The generalization of SINGLORAto non-square matrix preserves the stability and transformation-invariance properties demonstrated for the square case. *A proof is presented in the appendix.*

4 ON THE EXPRESSIVENESS OF SINGLORAIN TRANSFORMER ARCHITECTURES

In the previous section, we demonstrated that SINGLORAimproves the optimization process and yields more stable training dynamics. A natural concern that arises with SINGLORA's symmetric parameterization is whether it might limit the model's expressiveness compared to the general low-rank updates of standard LoRA. To address this, we now turn to investigating its expressive capacity within the Transformer architecture , which serves as the foundation of most current NLP and vision models. We analyze how SINGLORA's symmetric updates affect the key-query interaction in self-attention layers. For input $\mathbf{X} \in \mathbb{R}^{L\times d}$, the attention mechanism is computed as

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V},$$
$$\text{where } \mathbf{Q} = \mathbf{X}\mathbf{W}_q,\ \mathbf{K} = \mathbf{X}\mathbf{W}_k \text{ and } \mathbf{V} = \mathbf{X}\mathbf{W}_v.$$

When applying SINGLORA, the weight matrices become,

$$\mathbf{W}_q = \mathbf{W}_q^0 + \mathbf{A}_q\mathbf{A}_q^T \text{ and}$$
$$\mathbf{W}_k = \mathbf{W}_k^0 + \mathbf{A}_k\mathbf{A}_k^T$$

Examining the key-query interaction $\mathbf{Q}\mathbf{K}^T$, we get

$$\begin{aligned}\mathbf{Q}\mathbf{K}^T = \mathbf{X}[&\mathbf{W}_q^0\mathbf{W}_k^{0T} + \mathbf{W}_q^0\mathbf{A}_k\mathbf{A}_k^T \\ &+ \mathbf{A}_q\mathbf{A}_q^T\mathbf{W}_k^{0T} + \mathbf{A}_q\mathbf{A}_q^T\mathbf{A}_k\mathbf{A}_k^T]\mathbf{X}^T.\end{aligned}$$

The crucial insight that emerges is that although SINGLORAuses symmetric updates ($\mathbf{A}_q\mathbf{A}_q^T$ and $\mathbf{A}_k\mathbf{A}_k^T$), their interaction in the computation of attention is more general. The product of two symmetric matrices $(\mathbf{A}_q\mathbf{A}_q^T)(\mathbf{A}_k\mathbf{A}_k^T)$ is not necessarily symmetric unless they commute. Since there is no constraint forcing $\mathbf{A}_q\mathbf{A}_q^T$ and $\mathbf{A}_k\mathbf{A}_k^T$ to commute, SINGLORAcan learn general (non-symmetric) transformations of attention patterns despite its symmetric parameterization and does not fundamentally limit the model's ability to learn diverse attention patterns.

Method	LoRA	LoRA+	DoRA	SINGLORA
Accuracy	89.1	90.2	90.6	**91.3**
# Params	20M	20M	21M	12M

Table 2: Accuracy of Llama tuned on MNLI with SINGLORAand baselines with ranks 8.

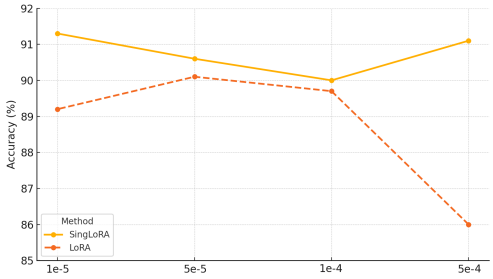

Figure 2: Accuracy of Llama-7B fine-tuned on MNLI across different learning rates. The plot compares the stability of LoRA and SINGLORAunder varying learning rates, demonstrating that SINGLORA's accuracy fluctuates by approximately 1%, while LoRA's performance varies by 4.8%. These results highlight the robustness of SINGLORA's optimization dynamics.

Synthetic experiment. To empirically validate this property, we implement LoRA and SINGLORAin an attention mechanism, where they learn to approximate a target attention pattern $Z \in \mathbb{R}^{128\times128}$ given input $X \in \mathbb{R}^{128\times128}$. Attention scores are computed according to 6. For SINGLORA, we define $W_q = W_q^0 + A_q A_q^\top$ and $W_k = W_k^0 + A_k A_k^\top$, where $W_q, W_k \in \mathbb{R}^{128\times128}$ simulates pretrained weights and $A_q, A_k \in \mathbb{R}^{128\times8}$ the adapter's matrices (with a rank equals to 8) of the query and keys projections.

For LoRA, we define $\hat{W}_q = \hat{W}_q^0 + \hat{B}_q\hat{A}_q$ and $\hat{W}_k = \hat{W}_k^0 + \hat{B}_k\hat{A}_k$, also with $\hat{W}_q, \hat{W}_k \in \mathbb{R}^{128\times128}$. Both approaches are configured to use the exact same number of parameters for a fair comparison. We optimize both models using an identical AdamW configuration with a learning rate of 10^{-4} for $7,000$ iterations, minimizing the losses $\|XW_qW_k^TX^T - Z\|_2^2$ and $\|X\hat{W}_q\hat{W}_k^TX^T - Z\|_2^2$ for SINGLORAand LoRA respectively.

To assess the stability and effectiveness of SINGLORA, we performed experiments in 1,000 different random initializations of Z and X. Figure 1 presents the average convergence behavior over these runs. We observe that SINGLORAconsistently outperforms LoRA in both convergence speed and final approximation error: SINGLORAreaches an error of approximately 10^{-5}, while LoRA plateaus around 10^{-2}.

These results highlight SINGLORA's ability to preserve expressiveness in attention mechanisms while significantly improving the optimization dynamics.

5 EXPERIMENTS

We conduct extensive experiments to evaluate SINGLORAin low-rank adaptation for linguistic and visual tasks. To evaluate SINGLORAin natural language processing, we consider selected comprehension and reasoning tasks from the **G**eneral **L**anguage **U**nderstanding **E**valuation (GLUE) benchmark (18). We strictly follow the NLP experimental protocol of LoRA+ (4), the leading approach for addressing LoRA's training stability issues, which thus serves as a key benchmark for SINGLORA. We also compare to LoRADoneRite (), a recent paper also addressed LoRA's instability. To fairly quantify algorithmic differences rather than hyper-parameter tuning advantages, we adopt their training/evaluation codebase, model architectures, modified layers, optimization settings, and training duration. We also compare to DoRA (9), the current state-of-the-art variant of LoRA. **Open source implementations of SINGLORAare available in the HuggingFace's PEFT package and in multiple non-official git repositories such as *https://github.com/kyegomez/SingLoRA/tree/main***.

Figure 3: Qualitative comparison of LoRA, DoRA and SINGLORAon Dreambooth.

5.1 LANGUAGE MODELS

RoBERTa-base and GPT-2. We first evaluate each approach based on its ability to fine-tune smaller language models - RoBERTa-base and GPT-2 - on the MNLI, QQP, and QNLI tasks from the GLUE benchmark. Following the LoRA+ setup, we set $r = \alpha = 8$. We use $u(t) = \min(\frac{t}{10^3}, 1)$ where t is the training step. Table 1 summarizes the accuracy in these tasks. Accuracies reported for LoRA and LoRA+ are directly taken from the original paper of LoRA+ (4).

The results show that SINGLORAoutperforms LoRA, achieving a 0.9% mean accuracy improvement for RoBERTa and 1.1% for GPT-2. It also achieves slightly better performance than LoRA+ and DoRA, while using only half the number of trainable parameters of both baselines. Note that while LoRA and LoRA+ explore multiple learning rates (5 for LoRA and 25 for LoRA+) and report results using carefully selected learning rates for each dataset and model, SINGLORAemploys a single learning rate across all experiments. This indicates that our method's stability reduces the need for extensive hyperparameter tuning, including the exhaustive grid search required for LoRA+. We further analyze the robustness to learning rate variations in ablation studies.

Llama 7B. To further validate our approach, we fine-tune a large language model (LLM), LLaMA-7B, on the MNLI task. Table 2 shows that SINGLORAoutperforms LoRA (by more than 2%), LoRA + (by more than 1%) and DoRA, while reducing the number of training parameters by 40%. Since fine-tuning LLMs such as LLaMA is one of the most common applications of low-rank adapters, this result underlines the practical advantages of our approach.

5.2 IMAGE GENERATION

To showcase the versatility of SINGLORAin architectures and modalities, we evaluated its effectiveness within diffusion models, where LoRA is commonly used to fine-tune models in small-scale datasets of specific subjects. We consider Stable Diffusion V1.5 (13), a popular image diffusion model.

Dreambooth. We benchmark approaches on DreamBooth (14), a known dataset with 30 classes of objects and animals, each containing 4–5 training images and 25 evaluation prompts. Following standard practice, we train each model for 400 iterations using the template prompt "`a photo of a sks <class>`", where "`sks`" is a rare English token. This setup allows the model to learn new object representations while retaining its general capacities. We finetune the query and key projections of the attention matrices in the U-Net component of Stable Diffusion. For all methods, training is conducted for 400 epochs using a learning rate of 10^{-3}. Figure 3 presents a qualitative comparison between the methods, illustrating how our adaptation approach enhances subject learn-

Method	CLIP Image	CLIP Text	DINO Similarity	# Params
LoRA	0.677	0.319	0.143	0.9M
LoRA+	0.688	0.315	0.150	0.9M
DoRA	0.687	0.317	0.148	0.9M
SingLoRA	0.690	0.317	0.151	0.9M

Table 3: Performance of Stable diffusion V5 finetuned in Dreambooth with the same number of parameters. Similarity of the generated image with the originals was measured using CLIP Image and DINO Similarity. Prompt fidelity is evaluated with CLIP Text.

Method	LoRA	DoRA	SingLoRA
DINO Similarity	0.463	0.471	0.501

Table 4: DINO Similarity of Stable Diffusion v1.5 tuned on a specific face with the reference face. SingLoRAand baselines uses the same number of trainable parameters.

ing. For instance, note that the shoe in the second row retains its iridescent color, whereas other methods fail to do so. Quantitatively, Table 3 shows that SingLoRAachieves a 5.4% improvement in the DINO (1) similarity score compared to DoRA, indicating a better object similarity, while maintaining prompt fidelity measured by the CLIP (11) Text score.

Human faces. To further analyze the ability of SingLoRAin visual tasks, we follow (22) and benchmark the adapters on a dataset that includes 40 human faces. This experiment quantifies the expressive power of adapters and their ability to learn complex details present in human faces. Samples of the dataset are available in the appendix. Following standard practice, we train each adapter for 1500 iterations using the template prompt "`a photo of a sks human`", where `sks` is a rare English token. Each adapter has the same number of trainable parameters. We finetune the query and key projections of the attention matrices in the U-Net component of Stable Diffusion. Table 4 shows an improvement in the DINO similarity score compared to LoRA and DoRA, indicating a better similarity to the reference image.

5.3 Ablation

Stability of SingLoRA. To validate the optimization stability analysis of SingLoRA, we compare its performance against LoRA in a range of learning rates. Specifically, we fine-tune Llama-7B in MNLI using learning rates ranging from $5 \cdot 10^{-5}$ to 10^{-4}. As shown in 2, the precision of SingLoRAfluctuates by approximately 1%, whereas LoRA exhibits a larger variation of up to 4.8%. These empirical results validate our theoretical findings, demonstrating that the design of SingLoRAinherently improves learning stability. In practice, this stability translates to simpler hyperparameter tuning, as our method maintains strong performance without requiring extensive searches for an optimal learning rate. Together with previous experiments, these findings highlight that SingLoRAnot only offers a more efficient and accurate parameterization, but also ensures robust convergence in practical settings.

6 Conclusion

We introduced SingLoRA, a novel formulation of low-rank adaptation that learns and employs a single matrix instead of two. Through theoretical analysis, we demonstrated that the proposed design inherently eliminates the inter-matrix scale disparities present in LoRA and guarantees stable feature learning without requiring special optimizers or careful hyperparameter tuning. Extensive experiments on language and vision tasks validated these benefits, consistently demonstrating improved performance with fewer trainable parameters than LoRA and its variants. Since our approach is complementary to various LoRA's variants, suchs as DoRA (9), a promising directi on is to explore their integration, harnessing their independent strengths to further enhance efficiency and performance.

REFERENCES

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[2] Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Krona: Parameter efficient tuning with kronecker adapter. *arXiv preprint arXiv:2212.10650*, 2022.

[3] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the impact of the activation function on deep neural networks training. In *International conference on machine learning*, pages 2672–2680. PMLR, 2019.

[4] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. *International Conference on Machine Learning*, 2024.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *International conference on computer vision (CVPR)*, 2015.

[6] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[7] Nam Hyeon-Woo, Moon Ye-Bin, and Tae-Hyun Oh. Fedpara: Low-rank hadamard product for communication-efficient federated learning. *International Conference on Learning Representations*, 2021.

[8] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *International Conference on Machine Learning*, 2024.

[10] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations*, 2019.

[11] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[12] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, 1951.

[13] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[14] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510, 2023.

[15] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

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

[17] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *Findings of the Association for Computational Linguistics*, 2022.

[18] Alex Wang. Glue: A multi-task benchmark and analysis platform for natural language understanding. *International Conference on Learning Representations*, 2018.

[19] Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

[20] Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite-depth neural networks. *arXiv preprint arXiv:2310.02244*, 2023.

[21] Jui-Nan Yen, Si Si, Zhao Meng, Felix Yu, Sai Surya Duvvuri, Inderjit S Dhillon, Cho-Jui Hsieh, and Sanjiv Kumar. Lora done rite: Robust invariant transformation equilibration for lora optimization. *International Conference on Learning Representations*, 2025.

[22] Fangzhao Zhang and Mert Pilanci. Riemannian preconditioned lora for fine-tuning foundation models. *arXiv preprint arXiv:2402.02347*, 2024.

[23] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *International Conference on Learning Representations*, 2023.

[24] Hongyun Zhou, Xiangyu Lu, Wang Xu, Conghui Zhu, Tiejun Zhao, and Muyun Yang. Lora-drop: Efficient lora parameter pruning based on output evaluation. *arXiv preprint arXiv:2402.07721*, 2024.

[25] Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices. *arXiv preprint arXiv:2309.02411*, 2023.

7 APPENDIX

7.1 PROOFS

In this section we demonstrate the theorem introduced in Sections 4 and 5. We rely on the following sufficient conditions for transformation invariance (see [20]),

$$\begin{array}{lrcl} \textbf{(i)} & \delta A_1 B_1 & = & \delta A_2 B_2 \\ \textbf{(ii)} & A_1 \delta B_1 & = & A_2 \delta B_2 \\ \textbf{(iii)} & \delta A_1 \delta B_1 & = & \delta A_2 \delta B_2 \end{array} \tag{6}$$

7.1.1 THEOREM 1

Any transformation invariant optimizer applying the same update rule for A and B achieves efficient feature learning.

Here for completeness we present the proof proposed in [20]. **Proof:**

Let $\|\mathbf{A}_1\| = \Theta(n^a)$, $\|\mathbf{B}_1\| = \Theta(n^b)$, $\|\nabla \mathbf{Z}\| = \Theta(n^c)$, and $\eta = \Theta(n^d)$, where η is the learning rate and n denotes the network width. Since $\mathbf{Z} = \mathbf{A}_1 \mathbf{B}_1^\top$, by the chain rule, we have $\nabla \mathbf{Z} = \nabla \mathbf{Z}^\top = \nabla \mathbf{Z} \mathbf{B}_1 \mathbf{A}_1^\top$. Given the symmetry of the update rule, the updates satisfy:

$$\|\delta \mathbf{A}_1\| = \Theta(n^{x+a+(y+1)b+c+d}), \quad \|\delta \mathbf{B}_1\| = \Theta(n^{x+b+(y+1)a+c+d}).$$

Assuming the update rule is invariant under scalar reparameterization, we compare two equivalent decompositions $\mathbf{A}_2 = n^s \mathbf{A}_1$ and $\mathbf{B}_2 = n^{-s} \mathbf{B}_1$, giving:

$$\|\delta \mathbf{A}_1\| \|\mathbf{B}_1\| = \|\delta \mathbf{A}_2\| \|\mathbf{B}_2\|.$$

From this, it follows:

$$xa + (y+1)b + zc + d = x(a+s) + (y+1)(b-s) + zc + d,$$

which simplifies to:

$$xs - (y+1)s = 0 \quad \forall s \Rightarrow x = -1.$$

Hence, we deduce:

$$\|\delta \mathbf{A}_1\| \|\mathbf{B}_1\| = \Theta(n^{a+(y+1)b+c+d}) = \Theta(n^{a+b+yc+d}).$$

Similarly, we find:

$$\|\mathbf{A}_1\| \|\delta \mathbf{B}_1\| = \Theta(n^{a+(y+1)b+c+d}) = \Theta(n^{a+b+yc+d}).$$

Given that these expressions are equal, the update process enables efficient feature learning:

$$\|\delta \mathbf{A}_1\| \|\mathbf{B}_1\| = \|\mathbf{A}_1\| \|\delta \mathbf{B}_1\| = \Theta(1),$$

by selecting a proper learning rate $\eta = \Theta(n^d)$, where $x = -1$ is fixed and d is chosen accordingly.

7.1.2 THEOREM 2

A gradient descent optimizer is transformation-invariant for SINGLORA.

Proof:

We prove that the three sufficient conditions for transformation invariance hold. **Proof of (i):** Assume two parametrizations of a LoRA adapter, A_1, $A_2 \in R^{n \times r}$ with identical ranks such that $A_1 A_1^\top = A_2 A_2^\top$. From the Polar Decomposition Theorem, there exists an orthogonal matrix $Q \in \mathbb{R}^{r \times r}$ such that $A_1 = A_2 Q$. Therefore, defining $X = A_1 A_1^\top$ and using the chain rule lead to,

$$\begin{aligned} \delta A_1 A_1^\top &= -\eta \nabla A_1 A_1^\top = -2\eta \nabla Z A_1 A_1^\top = -2\eta \nabla Z A_2 Q Q^\top A_2^\top \\ &= -2\eta \nabla Z A_2 A_2^\top = -\eta \nabla A_2 A_2^\top = \delta A_2 A_2^\top. \end{aligned} \tag{7}$$

The first sufficient condition for transformation invariance 6 is therefore satisfied.

Condition **(ii)** holds by symmetry with the condition **(i)**.

Proof of (iii): $\delta A_1\, \delta A_1 = \delta A_2\, \delta A_2$ For the gradients with respect to A, the chain rule gives

$$\nabla_{A_1}\mathcal{L} = \nabla_Z \mathcal{L} \cdot A_1 \quad \nabla_{A_2}\mathcal{L} = \nabla_Z \mathcal{L} \cdot A_2 \tag{8}$$

where $Z = A_i A_i^\top$ is the output of the adapted layer. Using the former relation and the Polar Theorem, there exists a matrix $Q \in \mathbb{R}^{\diagdown \times \diagdown}$ such that,

$$\nabla_{A_1}\mathcal{L} = \nabla_Z \mathcal{L}\, A_1 = (\nabla_Z \mathcal{L}\, A_2)Q = \nabla_{A_2}\mathcal{L}\, Q \tag{9}$$

Hence, the update for A becomes

$$\delta A_1 = -\eta\, \nabla_{A_1}\mathcal{L} = -\eta\, \nabla_{A_2}\mathcal{L}\, Q = \delta A_2\, Q. \tag{10}$$

Now, consider the product of the updates:

$$\begin{aligned} \delta A_1\, \delta A_1^\top &= (\delta A_2\, Q)(\delta A_2\, Q)^\top = \delta A_2\, (QQ^\top)\, \delta A_2^\top \\ &= \delta A_2\, \delta A_2^\top, \end{aligned} \tag{11}$$

since $QQ^\top = I$. This completes the proof of **(ii)**.

7.1.3 THEOREM 3

The generalization of SINGLORAto non-square matrix preserves the stability and transformation-invariance properties demonstrated for the square case.

Proof:

Recall that for a rectangular weight matrix,

$$W_0 \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}} \quad (d_{\text{in}} < d_{\text{out}}),$$

a low-rank adapter is defined via a matrix $A \in \mathbb{R}^{d_{\text{out}} \times r}$, with its truncation $A^* \in \mathbb{R}^{d_{\text{in}} \times r}$ formed by the first d_{in} rows of A.

Suppose that there exist two matrices $A_1, A_2 \in \mathbb{R}^{d_{\text{out}} \times r}$ such that their truncations satisfy

$$A_1^* A_1^\top = A_2^* A_2^\top.$$

For clarity, divide each A_i as

$$A_i = \begin{bmatrix} X_i \\ Y_i \end{bmatrix}, \quad i = 1, 2,$$

where $X_i \in \mathbb{R}^{d_{\text{in}} \times r}$ and $Y_i \in \mathbb{R}^{(d_{\text{out}} - d_{\text{in}}) \times r}$.

By definition, the equality $A_1^* A_1^\top = A_2^* A_2^\top$ implies,

$$\begin{aligned} X_1 X_1^\top &= X_2 X_2^\top \\ X_1 Y_1^\top &= X_2 Y_2^\top. \end{aligned} \tag{12}$$

Using the polar theorem, X_2 and Y_2 admit a polar decomposition:

$$\begin{aligned} X_2 &= X_1 Q \\ Y_2 &= Y_1 Q. \end{aligned} \tag{13}$$

Where Q is a symmetric orthogonal matrix.

Denote the gradient descent updates for A_i as,

$$\delta A_i = \begin{bmatrix} \delta X_i \\ \delta Y_i \end{bmatrix}, \quad i = 1, 2.$$

By equation equation 13 we have,

$$\delta A_2 = \begin{bmatrix} \delta X_1 \, Q \\ \delta Y_1 \, Q \end{bmatrix}. \tag{14}$$

We now demonstrate that the three sufficient conditions for transformation invariance hold. **Proof of (i) and (ii):** $A_1 \, \delta A_1^\top = A_2 \, \delta A_2^\top$**.** Using equations equation 13 and equation 14, we get

$$\begin{aligned} A_2^* \, \delta A_2^\top &= X_1 Q \begin{bmatrix} \delta X_1 Q \\ \delta Y_1 Q \end{bmatrix}^\top \\ &= [X_1 Q Q^\top \delta X_1^\top | X_1 Q Q^\top \delta Y_1^\top] \\ &= [X_1 \delta X_1^\top | X_1 \delta Y_1^\top] = X_1 \begin{bmatrix} \delta X_1 \\ \delta Y_1 \end{bmatrix}^\top \\ &= A_1^* \delta A_1^\top \end{aligned} \tag{15}$$

The proof for $A_1^{*\top} \delta A_1 = A_2^{*\top} \, \delta A_2$ is symmetric.

Proof of (iii): $\delta A_1 \, \delta A_1^\top = \delta A_2 \, \delta A_2^\top$**.** Using equations equation 13 and equation 14, we get

$$\begin{aligned} \delta A_2^* \, \delta A_2^\top &= \delta X_1 Q \begin{bmatrix} \delta X_1 Q \\ \delta Y_1 Q \end{bmatrix}^\top \\ &= [\delta X_1 Q Q^\top \delta X_1^\top | \delta X_1 Q Q^\top \delta Y_1^\top] \\ &= [\delta X_1 \delta X_1^\top | \delta X_1 \delta Y_1^\top] = \\ &= \delta A_1^* \delta A_1^\top \end{aligned} \tag{16}$$

This completes the proof of **iii**.

7.2 HARDWARE

All experiments were performed on a single NVIDIA A40 GPU with a memory of 48GB.

Figure 4: Samples from the dataset used in our experiment. The dataset was automatically generated with a state-of-the-art image generation model trained on faces available in `https://this-person-does-not-exist.com/en`.