Initialization using Update Approximation is a *Silver Bullet* for Extremely Efficient Low-Rank Fine-Tuning

Kaustubh Ponkshe [*] [1] **Raghav Singhal** [*] [1] **Eduard Gorbunov** [1] **Alexey Tumanov** [2]
Samuel Horvath [1] **Praneeth Vepakomma** [1] [3]

Abstract

Low-rank adapters have become a standard approach for efficiently fine-tuning large language models (LLMs), but they often fall short of achieving the performance of full fine-tuning. We propose a method, **LoRA** **S**ilver **B**ullet or **LoRA-SB**, that approximates full fine-tuning within low-rank subspaces using a carefully designed initialization strategy. We theoretically demonstrate that the architecture of LoRA-XS—which inserts a trainable $r \times r$ matrix between B and A while keeping other matrices fixed—provides the precise conditions needed for this approximation. We leverage its constrained update space to achieve optimal scaling for high-rank gradient updates while removing the need for hyperparameter tuning. We prove that our initialization offers an optimal low-rank approximation of the initial gradient and preserves update directions throughout training. Extensive experiments across mathematical reasoning, commonsense reasoning, and language understanding tasks demonstrate that our approach exceeds the performance of standard LoRA while using **27-90x** fewer parameters, and comprehensively outperforms LoRA-XS. Our findings establish that it is possible to simulate full fine-tuning in low-rank subspaces, and achieve significant efficiency gains without sacrificing performance. We have released our code publicly at `https://github.com/RaghavSinghal10/lora-sb`.

1. Introduction

Pre-trained language models have become central to natural language processing, achieving state-of-the-art performance across diverse tasks (Radford et al., 2021; Kirillov et al., 2023; Achiam et al., 2023). While these models excel at general-purpose capabilities (Bubeck et al., 2023; Hao et al., 2022), adapting them to specific downstream tasks often requires fine-tuning. Although in-context learning (Brown et al., 2020; Radford et al., 2019) has gained popularity for its simplicity, it falls short in both performance and efficiency compared to fine-tuning (Liu et al., 2022). At the same time, full fine-tuning, while highly effective, is computationally expensive and impractical at scale, highlighting the need for more efficient adaptation techniques.

Parameter-efficient fine-tuning (PEFT) has emerged as a crucial approach for adapting large language models (LLMs) while addressing computational constraints. While full fine-tuning (FFT) typically achieves optimal performance, it requires updating billions of parameters, making it computationally intensive and often impractical for most applications. Low-rank decomposition methods, beginning with LoRA (Hu et al., 2021), have shown particular promise by significantly reducing trainable parameters through learning low-rank updates. This has sparked numerous advances in low-rank methods that either enhance performance through better optimization techniques and initialization strategies, or improve parameter efficiency through structured matrices and adaptive rank selection (Zhang et al., 2023; Wang et al., 2024b;a). However, these methods still face fundamental trade-offs, they must either maintain a relatively large number of parameters to match FFT performance, or accept a performance degradation when pursuing extreme parameter efficiency (Hu et al., 2021; Ding et al., 2023; Wang et al., 2024b). This raises an important question: **can we design low-rank methods that maintain FFT-competitive performance while drastically reducing the parameter count beyond current approaches?**

Low-rank decomposition methods operate on a fundamental premise: fine-tuning requires learning only a low-rank update to the pre-trained weights. Some theoretical work extends this hypothesis, suggesting that methods like LoRA can learn any low-rank approximation of the full fine-tuning gradient. However, the gradients computed by these methods to update their trainable adapters do not inherently pos-

[*]Equal contribution. Order decided by coin toss. [1]Mohamed bin Zayed University of Artificial Intelligence [2]Georgia Institute of Technology [3]Massachusetts Institute of Technology. Correspondence to: Kaustubh Ponkshe <kaustubh.ponkshe@mbzuai.ac.ae>, Raghav Singhal <raghav.singhal@mbzuai.ac.ae>.

Preprint. In progress.

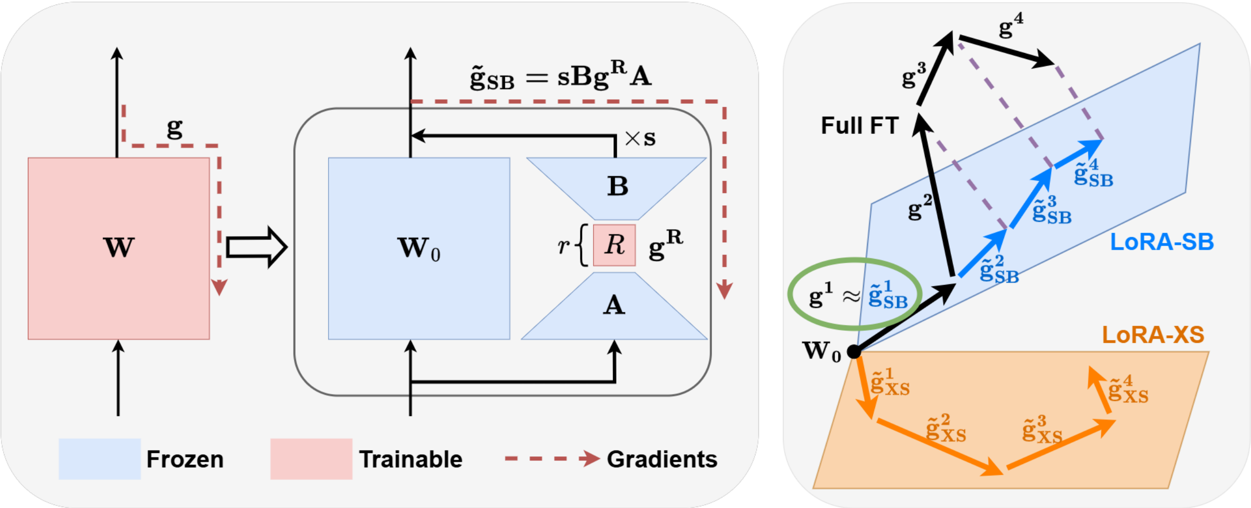

$\mathbf{g}$
$\mathbf{W}$
$\tilde{\mathbf{g}}_{\mathbf{SB}} = \mathbf{sBg^RA}$
$\times\mathbf{s}$
$\mathbf{W}_0$
$\mathbf{B}$
r
R
$\mathbf{g^R}$
$\mathbf{A}$
Frozen
Trainable
Gradients
$\mathbf{g^4}$
$\mathbf{g^3}$
Full FT
$\mathbf{g^2}$
$\tilde{\mathbf{g}}^4_{\mathbf{SB}}$
$\tilde{\mathbf{g}}^3_{\mathbf{SB}}$
LoRA-SB
$\tilde{\mathbf{g}}^2_{\mathbf{SB}}$
$\mathbf{g^1} \approx \tilde{\mathbf{g}}^1_{\mathbf{SB}}$
$\mathbf{W}_0$
LoRA-XS
$\tilde{\mathbf{g}}^1_{\mathbf{XS}}$
$\tilde{\mathbf{g}}^4_{\mathbf{XS}}$
$\tilde{\mathbf{g}}^2_{\mathbf{XS}}$
$\tilde{\mathbf{g}}^3_{\mathbf{XS}}$

Figure 1: **LoRA-SB Illustration.** LoRA-XS (Bałazy et al., 2024) reduces parameter count compared to LoRA (Hu et al., 2021) by inserting a trainable $r \times r$ matrix R between B and A, while keeping other matrices fixed, leading to $W = W_0 + sBRA$. Our method, LoRA-SB, leverages the same architecture. We find that updating R using its gradients g^R is equivalent to updating the full-finetuning matrix W with an equivalent gradient $\tilde{g}_{SB} = sBg^RA$. We initialize B, R, and A such that the equivalent gradient $\tilde{g}_{SB}$ optimally approximates the full fine-tuning gradient g in low rank subspaces **at each training step**. In essence, we simulate the **entire full fine-tuning process** optimally within low-rank subspaces by **utilizing only the first gradient** g_1 (shown in green) from full fine-tuning.

sess this property. For instance, LoRA's gradients need explicit optimization at each step to better approximate the full fine-tuning gradient (Wang et al., 2024b). Additionally, initialization has emerged as a critical factor in low-rank adaptation, as highlighted by recent works like Pissa-LoRA (Meng et al., 2024) and LoRA-GA (Wang et al., 2024a), which address this challenge through various low-rank approximation techniques.

We formally analyze these limitations in the context of the architecture used in LoRA-XS (Bałazy et al., 2024)—which inserts a trainable $r \times r$ matrix between B and A while keeping other matrices fixed—and demonstrate that these challenges are even more pronounced in its framework. While exploring solutions inspired by existing LoRA-based methods, we discover a remarkable property unique to LoRA-XS: through careful initialization of matrices A and B, we can approximately simulate the full fine-tuning optimization in low rank subspaces **throughout the entire training**, as shown in Figure 1. Our initialization strategy provides optimal scaling for approximating high-rank full fine-tuning gradients and eliminates the need for the scaling hyperparameter α, results which we prove theoretically. Our key contributions include:

- We theoretically formalize the limitations of LoRA-XS, showing how its constrained update space leads to suboptimal gradient approximation, initialization sensitivity, and hyperparameter dependence.

- We propose a principled initialization strategy derived from approximating the first step of full fine-tuning, proving it provides optimal low-rank approximation of the initial gradient and preserves update directions throughout training.

- We prove that our initialization makes gradient optimization hyperparameter-independent and guarantees convergence by maintaining orthonormal bases, eliminating the need for any tuning of the scaling factor.

- Through extensive experiments on 4 models across 16 datasets covering mathematical reasoning, commonsense reasoning, and language understanding tasks, we demonstrate that our method surpasses the performance of LoRA while using **27-90x** less parameters, and comprehensively outperforms LoRA-XS.

2. Related Work

Parameter-Efficient Fine-Tuning. Parameter-Efficient Fine-Tuning (PEFT) methods have become essential for adapting large pre-trained models under computational constraints. Early techniques like AdapterFusion (Pfeiffer et al., 2021) and Prefix-Tuning (Li & Liang, 2021) enabled task-specific adaptation with minimal parameter updates. Advances like soft prompts (Lester et al., 2021) further reduced trainable parameter counts while maintaining strong performance. Recent approaches have explored operating directly on model representations (Wu et al., 2024), offering different trade-offs between efficiency and performance.

Low-Rank Decomposition Methods. LoRA (Hu et al., 2021) demonstrated that weight updates during fine-tuning could be efficiently approximated using low-rank matrices, drastically reducing parameter counts while freezing pre-trained weights. Building on this insight, variants such as QLoRA (Dettmers et al., 2023) and AdaLoRA (Zhang et al., 2023) extended the paradigm through quantization and adaptive allocation strategies. The applicability of low-rank techniques has also been explored in pretraining with GaLore (Zhao et al., 2024) and ReLoRA (Lialin et al., 2023), highlighting the versatility of low-rank adaptation in both fine-tuning and pre-training contexts. LoRA-based methods have also been applied in other domains, such as efficient federated fine-tuning (Sun et al., 2024; Singhal et al., 2024).

Enhancing LoRA Performance. Recent efforts have focused on optimizing LoRA's performance. PiSSA-LoRA (Meng et al., 2024) demonstrated faster convergence and improved task performance by initializing matrices with principal components of pre-trained weights. LoRA-Pro (Wang et al., 2024b) and LoRA-GA (Wang et al., 2024a) improved gradient approximation, aligning low-rank updates more closely with full fine-tuning. Other methods like DoRA (Liu et al., 2024) and rsLoRA (Kalajdzievski, 2023) introduced decomposition-based and scaling stabilization techniques to enhance learning stability and expand LoRA's utility.

Improving Efficiency in LoRA Variants. Efficiency-focused innovations have pushed LoRA toward even greater parameter savings. LoRA-XS (Bałazy et al., 2024) achieves parameter reduction by inserting a small trainable weight matrix into frozen low-rank matrices. VeRA (Kopiczko et al., 2024) shares low-rank matrices across layers, relying on scaling vectors for task-specific adaptation. Tied-LoRA (Renduchintala et al., 2024) leverages weight tying to reduce parameter usage at higher ranks, while HydraLoRA (Tian et al., 2024) introduces an asymmetric architecture to improve adaptation in complex domains without domain-specific expertise.

3. Methodology

3.1. Preliminaries

In standard fine-tuning, a pre-trained weight matrix $W \in \mathbb{R}^{m\times n}$ is updated as:

$$W = W_0 + \Delta W \tag{1}$$

where W_0 is the pre-trained weight. This requires updating mn parameters per layer.

LoRA hypothesizes that these updates lie in a low-dimensional subspace and parameterizes the update matrix ΔW as:

$$W = W_0 + sBA \tag{2}$$

where $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$ are trainable low-rank matrices, with rank $r \ll \min(m, n)$, and s is a scaling factor to stabilize training. This reduces parameters from mn to $r(m + n)$.

LoRA-XS introduces a more efficient parameterization:

$$W = W_0 + sBRA \tag{3}$$

where $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$ are fixed matrices, and only $R \in \mathbb{R}^{r\times r}$ is trainable, reducing parameters to r^2.

We denote full fine-tuning gradient as $g = \frac{\partial L}{\partial W}$, and LoRA-XS gradient as $g^R_{\text{LoRA-XS}} = \frac{\partial L}{\partial R}$.

3.2. Motivation

LoRA-XS (Bałazy et al., 2024), although having significantly fewer learnable parameters compared to LoRA, exhibits suboptimal performance. The difference in architecture causes constraints on the type of updates LoRA-XS can learn. The subspace of learned updates is characterized in Lemma 1.

Lemma 1. *Let ΔW be an update learned by LoRA-XS . Then, the set of all ΔW, say $\mathcal{W}_{LoRA-XS}$, can be given by*

$$\{M \in \mathbb{R}^{m\times n} | Col(W) \subseteq Col(M) \\ \wedge Row(M) \subseteq Row(A)\}$$

Proof. Appendix A.2 □

The above lemma implies that while ΔW is constrained to be r ranked or lower, it also needs to have row and column spaces defined by those of B and A respectively. As against this, LoRA can learn any update ΔW as long as $\text{rank}(\Delta W) \leq r$. Thus the low expressivity of LoRA-XS as compared to LoRA can account for the performance drop. We identify three key limitations which arise due to this and otherwise:

1) **Inadequate Gradient Approximation:** LoRA optimization is mathematically equivalent to full fine-tuning using a constrained low-rank gradient. The gradient of LoRA does not optimally approximate the full gradient, and needs to be tuned at each step. LoRA-Pro (Wang et al., 2024b) finds that this results in suboptimal performances, and provides a closed form solution to optimize the gradients. In LoRA-XS , the gradient updates are restricted to an even more constrained low-rank space since the matrices A and B are

fixed. We posit that the limitation becomes particularly severe when the ideal updates lie outside the space spanned by fixed A and B, and consequently has a larger impact on performance.

2) **Suboptimal Initialization:** While initialization impacts all low-rank methods, it becomes critical in LoRA-XS where A and B are frozen. Unlike LoRA where poor initialization can be compensated through training, LoRA-XS must rely entirely on its initial subspace defined by A and B. Consider the zero initialization of the B matrix, for example. While LoRA may experience some performance degradation in this case (Wang et al., 2024a) (Meng et al., 2024), the ideal low-rank update ΔW can still be reached through gradient descent. In fact, zero initialization for the B matrix is a commonly used choice, including in the original LoRA implementation (Hu et al., 2021). However, in the case of LoRA-XS , this would result in no learning, as the product BRA would remain zero. The current LoRA-XS method leverages the most significant subspaces spanned by the columns of pre-trained weights W_0, inspired by (Meng et al., 2024). This initialization is inadequate because it fails to capture the specific subspaces relevant to the fine-tuning task. This issue, known to affect LoRA 's performance (Wang et al., 2024a), could have an even greater impact on LoRA-XS for the reasons discussed above.

3) **Hyperparameter Sensitivity:** The scaling factor α, present in almost every LoRA based fine-tuning method requires careful tuning to maintain stability during training. This hyperparameter acts as a bridge between the low-rank and full-rank spaces, compensating for the dimensional mismatch in gradients. Poor tuning of α can lead to unstable training or slow convergence (Kalajdzievski, 2023), adding complexity to the adaptation process and potentially limiting practical deployment.

3.3. Approximation of the full fine-tuning gradient

As mentioned above, LoRA optimization is mathematically equivalent to full fine-tuning using a constrained low-rank gradient. However the update generated using the gradients of LoRA does not result in the same update which the low-rank gradient would have generated. The following holds true for LoRA-XS as well.

To understand this, let us look at the change in weight W and its relationship with changing of low-rank matrix R. The relationship can be simply given by $dW = \frac{\partial W}{\partial R} g^R$. This relationship implies that updating matrix R with gradient g^R is equivalent to updating W with low rank equivalent gradient $\tilde{g}$ in full fine-tuning as described in Definition 1.

Definition 1 (Equivalent Gradient)**.** *In the context of LoRA-XS optimization, we define the equivalent gradient as:*

$$\tilde{g} = sBg^R A$$

where s is the scaling factor, and g^R is the gradient with respect to matrix R.

The equivalent gradient describes the virtual low-rank gradient of matrix W in LoRA-XS optimization process, despite W not being directly trainable. This gradient determines how updates to R affect the overall weight matrix. To bridge the performance gap between LoRA-XS and full fine-tuning, our goal is to minimize the discrepancy between the equivalent gradient $\tilde{g}$ and the full gradient g. First, we establish the relationship between gradients in LoRA-XS optimization in Lemma 2.

Lemma 2. *During LoRA-XS optimization, the gradient of the loss with respect to matrix R can be expressed in terms of the gradient with respect to the weight matrix W as:*

$$g^R_{LoRA-XS} = sB^T g A^T$$

Proof. Appendix A.2 □

Leveraging this relationship, we can formulate our objective to minimize the distance between the equivalent gradient and the full gradient. We do not have access to the full fine-tuning gradient g during LoRA-XS based fine-tuning. Thus we need to find the ideal gradient with respect to R, given by g^R, and subsequently the optimal approximation $\tilde{g}$, in terms of the gradient which is available to us during training: $g^R_{LoRA-XS}$. Fortunately, this optimization problem admits a closed-form solution independent of g as described in Theorem 3.

Theorem 3. *Assume matrices $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$ are both full rank. For the objective $min_{g^R} ||\tilde{g} - g||_F^2$, such that $\tilde{g} = sBg^R A$, the optimal solution is given by:*

$$g^R = \frac{1}{s^2}(B^T B)^{-1} g^R_{LoRA-XS} (AA^T)^{-1} \tag{4}$$

Proof. Appendix A.3. □

The closed-form solution presented in Theorem 3 solves the optimization problem $\min_{g^R} ||\tilde{g} - g||_F^2$, but by itself doesn't ensure the loss will decrease when updating R. Through Theorem 4, we prove that the change in loss is non-positive ($\mathrm{d}L \leq 0$). This property is fundamental to optimization as it guarantees consistent progress toward minimizing the loss throughout training.

Theorem 4. *Consider the update for matrix R using the closed-form solution derived in Theorem 3:*

$$R \leftarrow R - \eta g^R$$

where $\eta \geq 0$ is the learning rate. This update guarantees a reduction in the loss function, similar to traditional gradient descent methods. Specifically, the change in loss $\mathrm{d}L$ is given by:

$$\mathrm{d}L = -\eta \langle g^R_{LoRA-XS}, g^R \rangle \leq 0$$

Proof. Appendix A.4. □

3.4. Initialization using update approximation

Let us first consider the objective in fine-tuning, the primary goal is to update weights to better suit the target task and domain. The initial gradient steps are particularly informative, as they indicate the direction of desired adaptation. We propose leveraging this insight by using the first update step from full fine-tuning to initialize our low-rank matrices.

This approach offers two key advantages: First, it ensures the low-rank space captures the most relevant subspace for the target task rather than relying on pre-trained weight properties. Second, since A and B matrices will remain fixed in LoRA-XS, initializing them to span the subspace of early adaptation increases the likelihood of capturing useful updates throughout training. This would also ensure that the final update is learnt in the correct subspace, of which we have no apriori information besides the first step of full fine-tuning.

In essence, our method can be summarized as follows: Set an initialization which best approximates the first step of full fine-tuning. Formally, given a full fine-tuning update ΔW, our initialization should satisfy:

$$sB_{init}R_{init}A_{init} \approx \Delta W_{first-step} \tag{5}$$

The first step of full fine-tuning, for Adam-based optimizers such as AdamW, for a sample x_i is:

$$\Delta W_{first-step} = -\eta \times \textbf{sign}(\nabla_W \mathcal{L}(W_0, x_i)) \tag{6}$$

Using a single sample may lead to noisy estimates. Instead, we compute a more stable initialization by averaging gradients over a subset of the training data:

$$\Delta W_{avg} = -\eta\textbf{sign}(\sum_{i=0}^{n\leq|\mathbb{X}|} \nabla_W \mathcal{L}(W_0, x_i)) \ldots x_i \in \mathbb{X} \tag{7}$$

This averaged gradient better captures the general direction of adaptation required for the target task while being less sensitive to individual sample variations. We can then use truncated SVD to obtain a low-rank approximation of ΔW_{avg}, and express it as $sBRA$. There exist infinite combinations of B and A which can obey the above relationship. For instance, we can initialize B and A as US and V^T and keep R as I/s. This initialization is equivalent to the B and A initialization in LoRA-XS but by approximating the update rather than the pre-trained matrix. We note that the above process can be computed for any optimizer, by approximating the corresponding first step. We compute this specifically for AdamW since we use it.

3.5. Hyperparameter independence

The hyperparameter α is used in LoRA and subsequent decomposition based models to tackle the issue of instability caused to improper scaling of the updates. The gradient scaling is accounted for, by adding a hyperparameter to normalize the updates. The importance of scaling is well documented in methods like rank stabilization (Kalajdzievski, 2023). However, the full fine-tuning gradient g needs no such tuning. We claim that approximating the full fine-tuning gradient removes the need for introducing a scaling factor.

Theorem 5. *The equivalent gradient $\tilde{g}$ is hyperparameter s independent for $\tilde{g} = sBg^RA$, but not for $\tilde{g} = sBg^R_{LoRA-XS}A$*

Proof. Appendix A.5 □

The hyperparameter independence of the equivalent gradient eliminates the need for manual gradient scaling. Updates to weight matrix W depend solely on this gradient (modulo learning rate), making any additional scaling redundant.

This can be understood by examining the relationship with full fine-tuning gradient g. Since g is naturally scaled for optimal weight updates, and our method approximates g in a constrained subspace, the equivalent gradient inherits appropriate scaling automatically. Notably, this property is unique to our gradient approximation approach and does not hold for standard LoRA-XS training.

3.6. LoRA-SB: Update approximation initialization is a *silver bullet*

The solutions discussed above independently address the gradient approximation and initialization problems, while also providing hyperparameter independence. Our proposed method, LoRA-SB, elegantly combines these solutions through a simple initialization strategy.

Our initialization strategy is derived from approximating the first step of full fine-tuning:

$$U, S, V^T \leftarrow \textbf{SVD}(\Delta W_{avg}) \tag{8}$$

$$B_{init} \leftarrow U[1:r] \tag{9}$$

$$A_{init} \leftarrow V[1:r] \tag{10}$$

$$R_{init} \leftarrow \frac{1}{s} S[1:r, 1:r] \tag{11}$$

By the Eckart-Young theorem (Eckart & Young, 1936; Mirsky, 1960), this gives the optimal rank-r approximation of the initial full fine-tuning update. where U, S, V are obtained from truncated SVD of the averaged first update ΔW_{avg}.

This initialization leads to several key advantages that address the problems identified earlier.

Simplified Gradient Optimization. This initialization ensures B_{init} and A_{init} form orthonormal bases in $\mathbb{R}^m$ and $\mathbb{R}^n$ respectively, leading to $B^T B = AA^T = I$. With fixed B and A matrices being orthonormal, the optimal update step derived in Equation 3 simplifies to:

$$g^R = \frac{1}{s^2}(B^T B)^{-1} g^R_{LoRA-XS}(AA^T)^{-1} = \frac{1}{s^2} g^R_{LoRA-XS} \tag{12}$$

This eliminates the need for complex matrix inversions during training.

Optimum Update Approximation. Our initialization guarantees that the first update optimally approximates the full fine-tuning weight updates:

$$sB_{\text{init}} R_{\text{init}} A_{\text{init}} \approx \Delta W_{avg} \tag{13}$$

By the Eckart-Young theorem, this gives the optimal rank-r approximation of the initial full fine-tuning update.

Hyperparameter Independence. As shown in Theorem 5, when gradient approximation is applied with orthonormal B and A, the hyperparameter s can be set to 1, resulting in the following:

$$\boxed{g^R = g^R_{\text{LoRA-XS}}} \tag{14}$$

This demonstrates that our initialization guarantees optimal gradient approximation at every step, without requiring any scaling factor!

Guaranteed Loss Reduction. Since B is a tall orthonormal matrix and A is a wide orthonormal matrix, they remain full rank throughout training. This ensures that dL remains negative 4, guaranteeing stable optimization and convergence.

Another heuristic which might lead to a good initialization is setting the weights B and A, such that they match the first update also approximately matches the direction of ΔW.

$$\Delta(sB_{init} R_{init} A_{init}) \approx \gamma \Delta W \tag{15}$$

Thankfully, we don't have to choose between the two. For SGD, we prove that setting B_{init} and A_{init} using Equations 8-11, results in the first update of LoRA-XS to best approximate the direction of the update of full fine-tuning.

Theorem 6. *If A_{init} and B_{init} are initialized using LoRA-SB for the first SGD optimizer, then*

$$\Delta(B_{init} R_{init} A_{init}) \approx \Delta W$$

Proof. Appendix A.6 □

LoRA-SB Advantages over LoRA. Notably, many of the properties described above are not achievable with standard LoRA methods. Even if B and A are initialized as orthonormal in LoRA, subsequent updates do not preserve this property because B and A are trainable. This results in several challenges:

- Potential instability of $(B^T B)^{-1}$ and $(AA^T)^{-1}$, as they are not guaranteed to remain non-singular during training.
- Inability to ensure consistent loss reduction due to potential rank deficiency—B and A may not remain full-rank throughout training.
- Necessity to fine-tune the hyperparameter α.
- Repeated re-computation of $B^T B$ and AA^T is required at each optimizer step for accurate gradient approximation.

LoRA-SB's simple initialization strategy solves multiple problems simultaneously, offering a robust and efficient approach to low-rank adaptation.

Algorithm. We present a PyTorch-like implementation of our method in Algorithm 1. To optimize GPU memory usage during initialization, we hook into the backward pass of PyTorch and compute the gradient for one layer at a time, immediately discarding the computed gradients (Lv et al., 2024; Wang et al., 2024a). This ensures that memory usage remains at $O(1)$, independent of the number of layers. This approach keeps **memory consumption under control, ensuring it never exceeds the requirements of subsequent LoRA-SB fine-tuning**. For very large batch sizes, one can also use gradient accumulation and quantization to further manage memory usage within the specified limits.

4. Experiments

We evaluate our method over 16 different datasets on three widely-used NLP benchmarks, using models ranging from the 355 M-parameter RoBERTa-large model to the 9 B-parameter Gemma-2 model. Our setup spans both masked and autoregressive architectures, allowing us to comprehensively assess the effectiveness of our approach. We fine-tune

Algorithm 1 LoRA-SB, PyTorch-like

```
def initSB(model, D)
    # Estimate gradient with n samples
    $\Delta \mathbf{W}_{\text{avg}}$ ← est_grad(model, D, n)
    # Initialize B, R, A
    $\mathbf{B}, \mathbf{R}, \mathbf{A}$ ← trunc_SVD($\Delta \mathbf{W}_{\text{avg}}$)
    # Convert to LoRA-SB model
    sb_model ← lora_SB(model, B, R, A)
    return sb_model

# Load pre-trained model
model ← AutoModel(base_model)
# Initialize LoRA-SB with dataset D
sb_model ← initSB(model, D)
# Standard training, only R trainable
trainer ← Trainer(sb_model, ...)
trainer.train()
```

RoBERTa-large (Liu et al., 2019), Llama-3.2 3B (Dubey et al., 2024), Mistral-7B (Jiang et al., 2023), and Gemma-2 9B (Team et al., 2024), showcasing our method's adaptability across a variety of tasks and model architectures.

We implement all models using PyTorch (Paszke et al., 2019) and the popular HuggingFace Transformers library (Wolf et al., 2020). We run all experiments on a **single NVIDIA A6000 GPU** and report results as the average of three random seeds. To save memory, we initialize base models in `torch.bfloat16` precision. Appendix D provides detailed information on the datasets used.

We compute the update approximation using only $1/1000$ of each dataset's total number of samples. This ensures that the additional training time overhead is minimal and has a negligible effect on overall efficiency. We provide a benchmark of this overhead in Section 5. These samples are randomly selected from the dataset's training set in each run.

4.1. Arithmetic Reasoning

Details. We fine-tune Mistral-7B (Jiang et al., 2023) and Gemma-2 9B (Team et al., 2024) on 50K samples from the MetaMathQA (Yu et al., 2024) dataset and evaluate them on the GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) benchmarks. Evaluation focuses solely on the final numeric answer. We apply LoRA modules to the key, value, query, attention output, and all fully connected weight matrices, training with ranks $r = \{32, 64, 96\}$. Detailed hyperparameter settings are provided in Appendix C.

Results. We present the results for both models in Table 1. LoRA-SB significantly outperforms LoRA-XS across all parameter settings. Notably, LoRA-SB surpasses the performance of LoRA ($r = 32$) while using **40x** fewer trainable parameters for Mistral-7B and **90x** fewer for Gemma-2 9B at ranks $r = 96$ and $r = 64$, respectively.

We present training loss curves comparing LoRA-SB and LoRA-XS in Figure 2. Thanks to superior initialization, LoRA-SB starts with a lower initial loss compared to LoRA-XS. Additionally, due to optimal gradient approximation, LoRA-SB maintains a consistently better loss curve throughout training and converges to a superior final value.

4.2. Commonsense Reasoning

Details. We fine-tune Llama-3.2 3B (Dubey et al., 2024) on COMMONSENSE170K, a dataset that aggregates eight commonsense reasoning tasks (Hu et al., 2023). We evaluate the model's performance on each dataset individually, which include BoolQ (Clark et al., 2019), SIQA (Sap et al., 2019), PIQA (Bisk et al., 2020), ARC-Challenge (Clark et al., 2018), ARC-Easy (Clark et al., 2018), OBQA (Mihaylov et al., 2018), WinoGrande (Sakaguchi et al., 2021), and HellaSwag (Zellers et al., 2019). Each example is framed as a multiple-choice question where the model generates the correct answer without explanations. We use the prompt template from (Hu et al., 2023). LoRA modules are applied to the key, value, query, attention output, and all fully connected weight matrices, training with ranks $r = \{32, 64, 96\}$. Hyperparameter details are provided in Appendix C.

Results. We present the results in Table 2. LoRA-SB consistently outperforms LoRA-XS across all parameter settings. Additionally, LoRA-SB ($r = 96$) exceeds the performance of LoRA ($r = 32$) with **27x** fewer trainable parameters.

4.3. Natural Language Understanding

Details. We fine-tune RoBERTa-large (Liu et al., 2019) on GLUE, a sequence classification benchmark that contains several datasets, covering domains like sentiment analysis and natural language inference. The datasets we evaluate on are: CoLA (Warstadt et al., 2019), RTE, MRPC (Dolan & Brockett, 2005), SST-2 (Socher et al., 2013), QNLI (Rajpurkar et al., 2018), and STS-B (Cer et al., 2017). LoRA modules are applied only to the self-attention layers, following the configuration in the original LoRA paper (Hu et al., 2021), with ranks $r = \{8, 16, 24\}$. Detailed experimental settings are available in Appendix C.

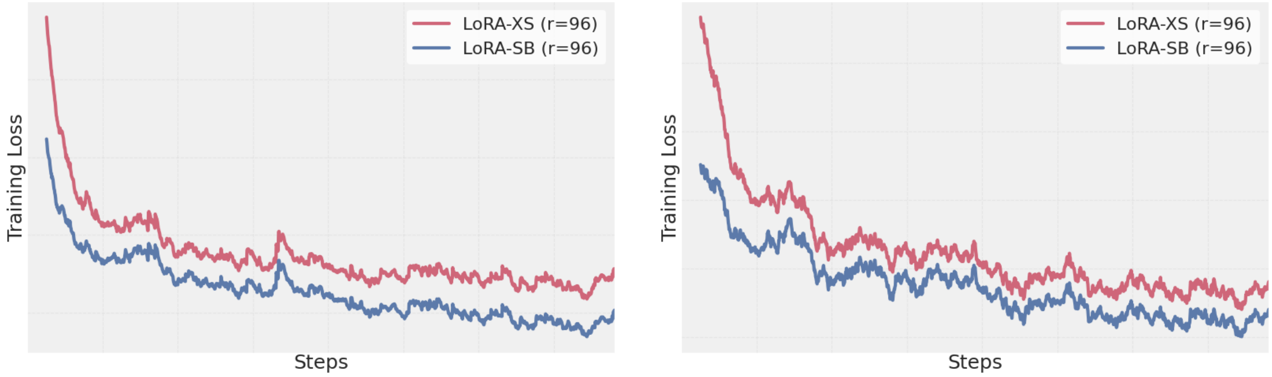
Results. The results are shown in Table 3. LoRA-SB consistently outperforms LoRA-XS across all parameter configurations. Notably, LoRA-SB ($r = 24$) outperforms LoRA ($r = 8$) with **39x** lesser trainable parameters.

5. Analysis

Optimal Initialization is Important!

Table 1: Accuracy comparison of fine-tuning methods on Mistral-7B and Gemma-2 9B across the arithmetic reasoning benchmarks GSM8K and MATH, after training on MetaMathQA. # Params denotes the number of trainable parameters. The best results among PEFT methods are highlighted in **bold**.

Model	**Method**	**Rank**	**# Params**	**Accuracy (↑)**	
				GSM8K	**MATH**
Mistral-7B	Full FT	-	7.24 B	63.87	17.65
	LoRA	32	83.88 M	61.94	15.98
	LoRA-XS	32	0.23 M	54.28	13.36
	LoRA-XS	64	0.92 M	57.08	15.62
	LoRA-XS	96	2.06 M	58.53	16.42
	LoRA-SB	32	0.23 M	58.91	15.28
	LoRA-SB	64	0.92 M	60.73	16.28
	LoRA-SB	96	2.06 M	**63.38**	**17.44**
Gemma-2 9B	Full FT	-	9.24 B	79.23	38.02
	LoRA	32	108.04 M	76.19	36.56
	LoRA-XS	32	0.30 M	74.07	34.62
	LoRA-XS	64	1.20 M	75.02	36.46
	LoRA-XS	96	2.71 M	75.21	36.98
	LoRA-SB	32	0.30 M	75.44	36.66
	LoRA-SB	64	1.20 M	76.65	37.14
	LoRA-SB	96	2.71 M	**78.40**	**37.70**



(a) Mistral-7B

(b) Gemma-2 9B

Figure 2: Training loss curves for Mistral-7B and Gemma-2 9B on MetaMathQA, comparing LoRA-SB and LoRA-XS.

Table 2: Accuracy comparison of fine-tuning methods on Llama-3.2 3B across eight commonsense reasoning datasets. # Params denotes the number of trainable parameters. The best results among PEFT methods are highlighted in **bold**.

Model	**Method**	**Rank**	**# Params**	**Accuracy (↑)**								
				BoolQ	**PIQA**	**SIQA**	**HellaS.**	**WinoG.**	**ARC-e**	**ARC-c**	**OBQA**	**Avg.**
Llama-3.2 3B	Full FT	-	3.21 B	70.43	85.64	80.45	91.92	85.02	88.52	75.29	81.88	82.39
	LoRA	32	48.63 M	70.03	**85.20**	79.12	90.71	82.24	86.91	74.32	**81.87**	81.44
	LoRA-XS	32	0.20 M	65.01	82.87	76.17	87.32	80.12	84.78	70.31	75.71	77.79
	LoRA-XS	64	0.80 M	66.53	83.12	77.98	88.53	81.76	85.15	72.04	77.14	79.03
	LoRA-XS	96	1.81 M	67.28	83.35	78.66	88.99	82.08	85.18	72.61	78.88	79.63
	LoRA-SB	32	0.20 M	66.33	84.06	78.91	89.04	81.37	86.62	72.44	76.97	79.47
	LoRA-SB	64	0.80 M	68.35	84.55	79.94	**91.68**	83.03	87.84	74.83	80.12	81.29
	LoRA-SB	96	1.81 M	**70.34**	84.76	**80.19**	91.62	**84.61**	**87.92**	**74.74**	81.20	**81.92**

Table 3: Comparison of fine-tuning methods on RoBERTa-large across the GLUE benchmark datasets. # Params denotes the number of trainable parameters. The best results among PEFT methods are highlighted in **bold**. We use Pearson correlation for STS-B, Matthew's correlation for CoLA, and accuracy for others. Higher is better for each metric.

Model	Method	Rank	# Params	CoLA	RTE	MRPC	STS-B	QNLI	SST-2	All
				Mcc ↑	Acc ↑	Acc ↑	Corr ↑	Acc ↑	Acc ↑	Avg. ↑
RoBERTa-large	Full FT	-	355.36 M	68.44	83.42	90.21	91.76	93.92	96.21	87.33
	LoRA	8	2162.69 K	68.02	82.98	90.05	91.43	93.42	95.98	86.98
	LoRA-XS	8	6.14 K	61.07	75.23	86.21	89.29	92.44	94.72	83.16
	LoRA-XS	16	24.57 K	63.32	79.06	86.28	90.36	93.89	95.76	84.70
	LoRA-XS	24	55.20 K	66.27	80.14	88.48	90.77	93.21	95.89	85.79
	LoRA-SB	8	6.14 K	63.57	78.43	88.72	90.59	92.95	95.07	84.88
	LoRA-SB	16	24.57 K	64.36	82.31	89.71	91.24	93.89	95.87	86.23
	LoRA-SB	24	55.20 K	**68.28**	**83.03**	**90.12**	**91.65**	**93.75**	**96.11**	**87.16**

To isolate the impact of initialization on training, we perform truncated SVD on various matrices, including Kaiming initialization and ΔW_{avg} with varying levels of added Gaussian noise, as shown in Table 4. By applying truncated SVD, we ensure optimal gradient approximation, leading to initialization matrices B_{init} and A_{init} that form orthonormal bases in $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively. This results in $B^T B = AA^T = I$, allowing us to isolate the effect of initialization. The results clearly demonstrate the significance of initialization—our approach consistently outperforms other variants.

Table 4: Comparison of initialization strategies using Mistral-7B on GSM8K and MATH. All methods ensure optimal gradient approximation, with differences arising solely from the initialization.

Initialization Method	Accuracy (↑)	
	GSM8K	MATH
trunc_SVD (Kaiming)	00.00	00.00
trunc_SVD ($\Delta W_{avg} + \mathcal{N}_{\mu=10^{-2}}$)	00.00	00.00
trunc_SVD ($\Delta W_{avg} + \mathcal{N}_{\mu=10^{-3}}$)	58.83	14.76
trunc_SVD ($\Delta W_{avg} + \mathcal{N}_{\mu=10^{-4}}$)	60.19	15.96
trunc_SVD ($\Delta W_{avg} + \mathcal{N}_{\mu=10^{-5}}$)	60.65	15.98
LoRA-SB; trunc_SVD (ΔW_{avg})	**63.38**	**17.44**

Optimal Gradient Approximation is Important!

We aim to examine the effect of optimal gradient approximation on training. Specifically, we want $B_{\text{init}} R_{\text{init}} A_{\text{init}} \approx \Delta W_{avg}$ without enforcing $B^T B = AA^T = I$. We achieve this through the following steps:

$$U, S, V^T \leftarrow \textbf{SVD}(\Delta W_{avg}) \tag{16}$$

$$B_{\text{init}} \leftarrow U[1:r]S[1:r, 1:r] \tag{17}$$

$$A_{\text{init}} \leftarrow V[1:r] \tag{18}$$

$$R_{\text{init}} \leftarrow I \tag{19}$$

This construction ensures that $B_{\text{init}} R_{\text{init}} A_{\text{init}} \approx \Delta W_{avg}$, but only $AA^T = I$, while $B^T B \neq I$. This setup is suboptimal for gradient approximation since we do not explicity use the closed-form solution derived in Theorem 3.

We compare the resulting loss curves against LoRA-SB (which uses optimal gradient approximation) for Mistral-7B on MetaMathQA, as shown in Figure 3. Although both methods start similarly due to effective initialization, LoRA-SB converges to significantly better values, demonstrating the advantage of optimal gradient approximation. Furthermore, LoRA-SB achieves higher accuracies on the GSM8K and MATH benchmarks, with scores of 63.38 and 17.44 compared to 55.87 and 12.74, respectively.

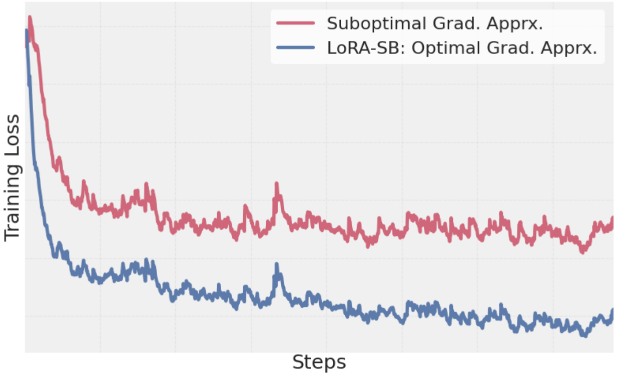


Figure 3: Training loss for Mistral-7B on MetaMathQA, highlighting the impact of optimal gradient approximation.

Training Time Overhead vs LoRA-XS

As previously mentioned, we compute the update approximation using only 1/1000 of the total training samples for each dataset. Table 5 presents the associated training time overhead for these computations, compared to LoRA-XS. The results show that the **additional overhead is negligible**, adding just 2–4 minutes compared to the total training time of 3–5 hours per epoch ($\approx 1.1\%$ to 1.3%). Additionally, the update computation is performed only once, at the beginning of the first epoch, prior to training.

Table 5: Training time overhead due to the initialization for various models on their respective tasks.

Model	Overhead	Training Time/Epoch
Mistral 7B	0:02:01	3:03:57
Gemma-2 9B	0:03:46	4:13:24
Llama-3.2 3B	0:03:54	4:54:31

Inference Overhead vs LoRA

LoRA-SB introduces a minimal inference cost overhead due to the insertion of the $r \times r$ matrix R between B and A, and the need for higher ranks to achieve comparable performance to LoRA. We benchmark the inference-time FLOPs and MACs across various models and find that the overhead is negligible. A detailed analysis can be found in Table 6 of Appendix B.

6. Conclusion

In this work, we introduced LoRA-SB, a novel framework that bridges the gap between low-rank parameter-efficient fine-tuning and full fine-tuning. This is enabled by our carefully designed initialization strategy, which approximates the first step of full fine-tuning and ensures that the most relevant subspaces for task-specific adaptation are captured from the outset. Our theoretical analysis uncovered critical limitations in LoRA-XS: suboptimal gradient approximation and reliance on fixed initialization that fails to adapt to task-specific nuances. By addressing both issues through a unified approach, we demonstrate that LoRA-SB achieves optimal scaling and preserves critical update directions. Furthermore, our approach ensures hyperparameter independence by directly approximating the full fine-tuning gradient, thereby eliminating instability and convergence issues associated with scaling factors.

Through extensive experiments on 4 models across 16 datasets covering mathematical reasoning, commonsense reasoning, and language understanding tasks, we demonstrate that our method exceeds the performance of LoRA while using upto **90x** less parameters, and comprehensively outperforms LoRA-XS. Our work advances the state of PEFT while laying the groundwork for further innovations in low-rank adaptations for neural networks. Future work includes exploring adaptive layer-wise rank settings and evaluating the approach on other architectures, such as Vision Language Models (VLMs) and Vision Transformers (ViTs).

7. Acknowledgements

This research was supported by funding from Mohamed bin Zayed University of Artificial Intelligence (MBZUAI) and ADIA Lab.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Bałazy, K., Banaei, M., Aberer, K., and Tabor, J. Lora-xs: Low-rank adaptation with extremely small number of parameters. (arXiv:2405.17604), October 2024. URL http://arxiv.org/abs/2405.17604. arXiv:2405.17604 [cs].

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., et al. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020. URL https://api.semanticscholar.org/CorpusID:218971783.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. (arXiv:2305.14314), May 2023. doi: 10.48550/arXiv.2305.14314. URL http://arxiv.org/abs/2305.14314. arXiv:2305.14314 [cs].

Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., Hu, S., Chen, Y., Chan, C.-M., Chen, W., Yi, J., Zhao, W., Wang, X., Liu, Z., Zheng, H.-T., Chen, J., Liu, Y., Tang, J., Li, J., and Sun, M. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, March 2023. ISSN 2522-5839. doi: 10.1038/s42256-023-00626-4.

Dolan, B. and Brockett, C. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*, 2005.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Hao, Y., Song, H., Dong, L., Huang, S., Chi, Z., Wang, W., Ma, S., and Wei, F. Language models are general-purpose interfaces. (arXiv:2206.06336), June 2022. doi: 10.48550/arXiv.2206.06336. URL http://arxiv.org/abs/2206.06336. arXiv:2206.06336 [cs].

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. (arXiv:2106.09685), October 2021. URL http://arxiv.org/abs/2106.09685. arXiv:2106.09685 [cs].

Hu, Z., Wang, L., Lan, Y., Xu, W., Lim, E.-P., Bing, L., Xu, X., Poria, S., and Lee, R. K.-W. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models, 2023. URL https://arxiv.org/abs/2304.01933.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Kalajdzievski, D. A rank stabilization scaling factor for fine-tuning with lora. (arXiv:2312.03732), November 2023. URL http://arxiv.org/abs/2312.03732. arXiv:2312.03732 [cs].

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., and Girshick, R. B. Segment anything. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3992–4003, 2023. URL https://api.semanticscholar.org/CorpusID:257952310.

Kopiczko, D. J., Blankevoort, T., and Asano, Y. M. Vera: Vector-based random matrix adaptation. (arXiv:2310.11454), January 2024. doi: 10.48550/arXiv.2310.11454. URL http://arxiv.org/abs/2310.11454. arXiv:2310.11454 [cs].

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. (arXiv:2104.08691), September 2021. doi: 10.48550/arXiv.2104.08691. URL http://arxiv.org/abs/2104.08691. arXiv:2104.08691 [cs].

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. (arXiv:2101.00190), January 2021. doi: 10.48550/arXiv.2101.00190. URL http://arxiv.org/abs/2101.00190. arXiv:2101.00190 [cs].

Lialin, V., Shivagunde, N., Muckatira, S., and Rumshisky, A. Relora: High-rank training through low-rank updates. (arXiv:2307.05695), December 2023. doi: 10.48550/arXiv.2307.05695. URL http://arxiv.org/abs/2307.05695. arXiv:2307.05695 [cs].

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. (arXiv:2205.05638), August 2022. doi: 10.48550/arXiv.2205.05638. URL http://arxiv.org/abs/2205.05638. arXiv:2205.05638 [cs].

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. Dora: Weight-decomposed low-rank adaptation. (arXiv:2402.09353), July 2024. doi: 10.48550/arXiv.2402.09353. URL http://arxiv.org/abs/2402.09353. arXiv:2402.09353 [cs].

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Lv, K., Yang, Y., Liu, T., Gao, Q., Guo, Q., and Qiu, X. Full parameter fine-tuning for large language models with limited resources, 2024. URL https://arxiv.org/abs/2306.09782.

Meng, F., Wang, Z., and Zhang, M. Pissa: Principal singular values and singular vectors adaptation of large language models. (arXiv:2404.02948), May 2024. URL http://arxiv.org/abs/2404.02948. arXiv:2404.02948 [cs].

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Mirsky, L. Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 11(1): 50–59, 1960.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., and Gurevych, I. Adapterfusion: Non-destructive task composition for transfer learning. (arXiv:2005.00247), January 2021. doi: 10.48550/arXiv.2005.00247. URL http://arxiv.org/abs/2005.00247. arXiv:2005.00247 [cs].

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:231591445.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

Renduchintala, A., Konuk, T., and Kuchaiev, O. Tied-lora: Enhancing parameter efficiency of lora with weight tying. (arXiv:2311.09578), April 2024. doi: 10.48550/arXiv.2311.09578. URL http://arxiv.org/abs/2311.09578. arXiv:2311.09578 [cs].

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Sap, M., Rashkin, H., Chen, D., LeBras, R., and Choi, Y. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

Singhal, R., Ponkshe, K., and Vepakomma, P. Exact aggregation for federated and efficient fine-tuning of foundation models, 2024. URL https://arxiv.org/abs/2410.09432.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Sun, Y., Li, Z., Li, Y., and Ding, B. Improving lora in privacy-preserving federated learning, 2024. URL https://arxiv.org/abs/2403.12313.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Tian, C., Shi, Z., Guo, Z., Li, L., and Xu, C. Hydralora: An asymmetric lora architecture for efficient fine-tuning. (arXiv:2404.19245), May 2024. doi: 10.48550/arXiv.2404.19245. URL http://arxiv.org/abs/2404.19245. arXiv:2404.19245 [cs].

Wang, S., Yu, L., and Li, J. Lora-ga: Low-rank adaptation with gradient approximation. (arXiv:2407.05000), July 2024a. URL http://arxiv.org/abs/2407.05000. arXiv:2407.05000 [cs].

Wang, Z., Liang, J., He, R., Wang, Z., and Tan, T. Lora-pro: Are low-rank adapters properly optimized? (arXiv:2407.18242), October 2024b. URL http://arxiv.org/abs/2407.18242. arXiv:2407.18242 [cs].

Warstadt, A., Singh, A., and Bowman, S. R. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019. doi: 10.1162/tacl_a_00290. URL https://aclanthology.org/Q19-1040.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Wu, Z., Arora, A., Wang, Z., Geiger, A., Jurafsky, D., Manning, C. D., and Potts, C. Reft: Representation finetuning for language models. (arXiv:2404.03592), May 2024. doi: 10.48550/arXiv.2404.03592. URL http://arxiv.org/abs/2404.03592. arXiv:2404.03592 [cs].

Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J. T., Li, Z., Weller, A., and Liu, W. Metamath: Bootstrap your own mathematical questions for large language models, 2024. URL https://arxiv.org/abs/2309.12284.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Zhang, Q., Chen, M., Bukharin, A., Karampatziakis, N., He, P., Cheng, Y., Chen, W., and Zhao, T. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. (arXiv:2303.10512), December 2023. doi: 10.48550/arXiv.2303.10512. URL http://arxiv.org/abs/2303.10512. arXiv:2303.10512 [cs].

Zhao, J., Zhang, Z., Chen, B., Wang, Z., Anandkumar, A., and Tian, Y. Galore: Memory-efficient llm training by gradient low-rank projection. (arXiv:2403.03507), June 2024. doi: 10.48550/arXiv.2403.03507. URL http://arxiv.org/abs/2403.03507. arXiv:2403.03507 [cs].

A. Mathematical Proofs

In all the proofs below, we will use the notations defined in Section 3.

A.1. Proof of Lemma 1

Lemma. *Let ΔW be an update learned by LoRA-XS . Then, the set of all ΔW, say $\mathcal{W}_{LoRA-XS}$, can be given by*

$$\mathcal{W}_{LoRA-XS} := \{M \in \mathbb{R}^{m\times n} | Col(M) \subseteq Col(B) \wedge \ Row(M) \subseteq Row(A)\}$$

Proof. The column space of ΔW is given by:

$$\text{Col}(\Delta W) = \{y \in \mathbb{R}^m \mid y = \Delta W x, \, x \in \mathbb{R}^n\} \tag{20}$$

Now since $\Delta W = BRA$:

$$\text{Col}(\Delta W) = \{y \in \mathbb{R}^m \mid y = BRAx, \, x \in \mathbb{R}^n\} \implies \text{Col}(\Delta W) = \{y \in \mathbb{R}^m \mid y = Bz, \, z \in \text{Col}(RA)\} \ldots \text{ for } z = RAx \tag{21}$$

Now:

$$\text{Col}(B) = \{y \in \mathbb{R}^m \mid y = Bx, \, x \in \mathbb{R}^r\} \tag{22}$$

Now since $\text{Col}(RA) \subseteq \mathbb{R}^r$:

$$\text{Col}(\Delta W) \subseteq \text{Col}(B) \tag{23}$$

A symmetric proof can now be given for row space of ΔW. □

A.2. Proof of Lemma 2

Lemma. *During LoRA-XS optimization, the gradient of the loss with respect to matrix R can be expressed in terms of the gradient with respect to the weight matrix W as:*

$$g^R_{LoRA-XS} = sB^T g A^T$$

Proof. Let L be the loss function. We have already defined g and $g^R_{\text{LoRA-XS}}$ as:

$$g := \frac{\partial L}{\partial W} \quad \& \quad g^R_{\text{LoRA-XS}} := \frac{\partial L}{\partial R} \tag{24}$$

Now by chain rule:

$$\frac{\partial L}{\partial R} = \frac{\partial L}{\partial W}\frac{\partial W}{\partial R} \implies \frac{\partial L}{\partial R} = \frac{\partial L}{\partial W}\frac{\partial W}{\partial X}\frac{\partial X}{\partial R} \ldots \text{ for } X = RA \tag{25}$$

We know that for $W = sBX$:

$$\frac{\partial L}{\partial W}\frac{\partial W}{\partial X} = sB^T g \implies \frac{\partial L}{\partial R} = sB^T g \frac{\partial X}{\partial R} \tag{26}$$

Let $sB^T g = y$. We know that when $X = RA$:

$$y\frac{\partial X}{\partial R} = yA^T \implies \frac{\partial L}{\partial R} = yA^T = sB^T g A^T \tag{27}$$

$$\text{Therefore,} \quad \boxed{g^R_{\text{LoRA-XS}} = sB^T g A^T} \tag{28}$$

□

A.3. Proof of Theorem 3

Theorem. *Assume matrices $B \in \mathbb{R}^{m\times r}$ and $A \in \mathbb{R}^{r\times n}$ are both full rank. For the objective min$_{g^R}||\tilde{g} - g||_F^2$, such that $\tilde{g} = sBg^R A$, the optimal solution is given by:*

$$g^R = \frac{1}{s^2}(B^T B)^{-1} g^R_{LoRA-XS}(AA^T)^{-1}$$

Proof. Since we already defined the equivalent gradient $\tilde{g} := sBg^R A$, the minimization problem can be formally denoted as:

$$\arg\min_{g^R} F = \|sBg^R A - g\|_F^2 \tag{29}$$

For differentiable F,

$$\frac{\partial F}{\partial g^R} = 0 \implies 2(\tilde{g} - g)\cdot\frac{\partial \tilde{g}}{\partial g^R} = 0 \implies 2(sBg^R A - g)\cdot\frac{\partial(sBg^R A)}{\partial g^R} = 0 \tag{30}$$

Using the same trick from before and substituting $g^R A = X$, we get:

$$2sB^T(sBg^R A - g)A^T = 0 \implies B^T(sBg^R A - g)A^T = 0 \implies B^T sBg^R AA^T = B^T gA^T \tag{31}$$

From Lemma 2, we get:

$$B^T gA^T = g^R_{\text{LoRA-XS}}/s \implies B^T sBg^R AA^T = g^R_{\text{LoRA-XS}}/s \implies B^T Bg^R AA^T = g^R_{\text{LoRA-XS}}/s^2 \tag{32}$$

Now since B and A are full rank, multiplying both sides by $(B^T B)^{-1}$ and $(AA^T)^{-1}$ on the left and right side respectively gives:

$$(B^T B)^{-1}(B^T Bg^R AA^T)(AA^T)^{-1} = (B^T B)^{-1} g^R_{\text{LoRA-XS}}(AA^T)^{-1}/s^2 \tag{33}$$

$$\text{Therefore,} \quad \boxed{g^R = \frac{1}{s^2}(B^T B)^{-1} g^R_{\text{LoRA-XS}}(AA^T)^{-1}} \tag{34}$$

□

A.4. Proof of Theorem 4

Theorem. *Consider the update for matrix R using the closed-form solution derived in Theorem 3:*

$$R \leftarrow R - \eta g^R$$

where $\eta \geq 0$ is the learning rate. This update guarantees a reduction in the loss function, similar to traditional gradient descent methods. Specifically, the change in loss $\mathrm{d}L$ is given by:

$$\mathrm{d}L = -\eta\langle g^R_{LoRA-XS}, g^R\rangle = -\eta\langle g^R_{LoRA-XS}, \frac{1}{s^2}(B^T B)^{-1} g^R_{LoRA-XS}(AA^T)^{-1}\rangle \leq 0$$

Proof. We establish that during optimization, the differential change in the loss function, $\mathrm{d}L$, can be expressed as:

$$\mathrm{d}L = -\eta\langle g^R_{\text{LoRA-XS}}, \frac{1}{s^2}(B^T B)^{-1} g^R_{\text{LoRA-XS}}(AA^T)^{-1}\rangle_F \tag{35}$$

Let us derive this expression step by step. First, the differential change in loss can be written as:

$$\mathrm{d}L = \langle \frac{\partial L}{\partial R}, \mathrm{d}R \rangle_F \tag{36}$$

The update equation for R follows:

$$\mathrm{d}R = -\eta g^R \tag{37}$$

Given that $\frac{\partial L}{\partial R} = g^R_{\text{LoRA-XS}}$, we obtain:

$$\begin{aligned} \mathrm{d}L &= -\eta \langle g^R_{\text{LoRA-XS}}, g^R \rangle_F \\ \mathrm{d}L &= -\eta \langle g^R_{\text{LoRA-XS}}, \frac{1}{s^2}(B^TB)^{-1} g^R_{\text{LoRA-XS}} (AA^T)^{-1} \rangle_F \end{aligned} \tag{38}$$

To prove that $\mathrm{d}L \leq 0$, we demonstrate that:

$$\langle g^R_{\text{LoRA-XS}}, \frac{1}{s^2}(B^TB)^{-1} g^R_{\text{LoRA-XS}} (AA^T)^{-1} \rangle_F \geq 0 \tag{39}$$

We first establish that $(B^TB)^{-1}$ is positive definite. For any non-zero vector x, since B is full-rank:

$$\langle x, B^TBx \rangle = \langle Bx, Bx \rangle = \|Bx\|^2 > 0 \tag{40}$$

This proves B^TB, and consequently, $(B^TB)^{-1}$ are positive definite. Through Cholesky decomposition, we can express $(B^TB)^{-1} = VV^T$. Similarly, $(AA^T)^{-1}$ is positive-definite and can be decomposed as $(AA^T)^{-1} = UU^T$.

Therefore:

$$\begin{aligned} \langle g^R_{\text{LoRA-XS}}, \frac{1}{s^2}(B^TB)^{-1} g^R_{\text{LoRA-XS}} (AA^T)^{-1} \rangle_F &= \frac{1}{s^2} \langle g^R_{\text{LoRA-XS}}, VV^T g^R_{\text{LoRA-XS}} UU^T \rangle_F \\ &= \frac{1}{s^2} \langle V^T g^R_{\text{LoRA-XS}} U, V^T g^R_{\text{LoRA-XS}} U \rangle_F \\ &= \frac{1}{s^2} \|V^T g^R_{\text{LoRA-XS}} U\|_F^2 \geq 0 \end{aligned} \tag{41}$$

Thus, we have proven:

$$\mathrm{d}L = -\eta \langle g^R_{\text{LoRA-XS}}, \frac{1}{s^2}(B^TB)^{-1} g^R_{\text{LoRA-XS}} (AA^T)^{-1} \rangle_F \leq 0 \tag{42}$$

For our specific initialization where $(B^TB) = I$, $(AA^T) = I$, and $s = 1$, the result simplifies to:

$$\mathrm{d}L = -\eta \langle g^R_{\text{LoRA-XS}}, g^R_{\text{LoRA-XS}} \rangle_F \leq 0 \tag{43}$$

□

A.5. Proof of Theorem 5

Theorem. *The equivalent gradient $\tilde{g}$ is hyperparameter s independent when*

$$\tilde{g} = sBg^RA \ldots \textit{ but not when } \tilde{g} = sBg^R_{LoRA-XS}A$$

Proof. Let g be the full fine-tuning gradient. We want to prove that $\tilde{g}$ does not depend on s, so we try to express it in terms of g which does not depend on the LoRA-XS training process or reparameterization

1) For $\tilde{g} = sBg^RA$:

$$g^R = \frac{1}{s^2}(B^TB)^{-1} g^R_{\text{LoRA-XS}} (AA^T)^{-1} \implies \tilde{g} = \frac{s}{s^2} B(B^TB^{-1}) g^R_{\text{LoRA-XS}} (AA^T)^{-1} A \tag{44}$$

Now since $g^R_{\text{LoRA-XS}} = sB^TgA^T$:

$$\tilde{g} = \frac{1}{s}B(B^TB^{-1})sB^TgA^T(AA^T)^{-1}A = B(B^TB^{-1})B^TgA^T(AA^T)^{-1}A \tag{45}$$

which is s independent!

2) For $\tilde{g} = sBg^R_{\text{LoRA-XS}}A$

$$g^R_{\text{LoRA-XS}} = sB^TgA^T \implies \tilde{g} = sB(sB^TgA^T)A \implies \tilde{g} = s^2BB^TgA^TA \tag{46}$$

which is not s independent! □

A.6. Proof of Theorem 6

Theorem. *If A_{init} and B_{init} are initialized using LoRA-SB for the first SGD optimizer, then*

$$\Delta(B_{init}R_{init}A_{init}) \approx \Delta W$$

Proof. Consider a gradient descent step with learning rate η and updates for R:

$$\Delta R = -\eta\nabla_R\mathcal{L}(R) \implies B\Delta RA = -\eta B\nabla_R\mathcal{L}(R)A \tag{47}$$

To measure its approximation quality of update of the weights in full finetuning:

$$\Delta W = -\eta\nabla_W\mathcal{L}(W_0) \tag{48}$$

We use Frobenius norm of the difference between these two updates as a criterion:

$$\|B\Delta RA - \eta\mathcal{L}_W(W_0)\|_F = \eta\|B\nabla_R\mathcal{L}(R)A - \mathcal{L}_W(W_0)\|_F \tag{49}$$

We have shown before that:

$$\nabla_R\mathcal{L} = B^T\nabla_W\mathcal{L}A^T \tag{50}$$

The problem becomes:

$$\min_{A_{\text{init}},B_{\text{init}}} \|B^T(B^T\nabla_W\mathcal{L}A^T)A - \nabla_W\mathcal{L}\|_F \ldots \text{ where } \nabla_W\mathcal{L} = USV^T \tag{51}$$

Using our initialization, we get:

$$\|BB^T\nabla_W\mathcal{L}A^TA - \nabla_W\mathcal{L}\|_F = \|U_{IR}U^T_{IR}UsV^TV_{IR}V^T_{IR} - UsV^T\|_F \tag{52}$$

Now:

$$U_{IR}U^T_{IR}UsV^TV_{IR}V^T_{IR} = \sum_{i=1}^{r}\sigma_iu_iv_i^T \tag{53}$$

Now rank of W' where:

$$W' = U_{IR}U^T_{IR}UsV^TV_{IR}V^T_{IR} \tag{54}$$

is $\leq r$ since rank of B_{init} and A_{init} is r

By Eckart-Young Theorem, the optimal low-rank optimization w.r.t Frobenius norm is:

$$W'^* = \underset{\text{rank}(W')=r}{\text{arg min}} \|W' - \nabla_W\mathcal{L}\|_F = \sum_{i=1}^{r}\sigma_iu_iv_i^T \tag{55}$$

Since we get an identical expression, our solution is optimal. □

B. Inference Costs

As discussed in Section 5, LoRA-SB introduces minimal inference cost overhead due to the insertion of the $r \times r$ matrix R between B and A, as well as the requirement for higher ranks to match LoRA's performance. This comparison is presented in Table 6, showing that the additional overhead of LoRA-SB is negligible.

Table 6: Inference cost comparison between LoRA-SB and LoRA across various models for a sequence length of 256. The minimum rank at which LoRA-SB matches or exceeds LoRA's performance is highlighted in **bold**.

Model	**Method**	**Rank**	**MACs**	**FLOPs**
RoBERTa-large	LoRA	8	77.86 G	155.79 G
	LoRA-SB	16	78.42 G	156.91 G
	LoRA-SB	24	78.97 G	158.01 G
LlaMA-3.2 3B	LoRA	32	0.84 T	1.67 T
	LoRA-SB	64	0.85 T	1.70 T
	LoRA-SB	96	0.86 T	1.72 T
Mistral 7B	LoRA	32	1.84 T	3.69 T
	LoRA-SB	64	1.86 T	3.73 T
	LoRA-SB	92	1.88 T	3.77 T
Gemma-2 9B	LoRA	32	3.89 T	7.77 T
	LoRA-SB	64	3.93 T	7.86 T
	LoRA-SB	96	3.97 T	7.94 T

C. Experiment Details

We ran our experiments on a single NVIDIA A6000 GPU, averaging results over three independent trials. We trained all models using the AdamW optimizer (Loshchilov & Hutter, 2019).

For arithmetic and commonsense reasoning tasks, we set up Mistral-7B, Gemma-2 9B, and Llama-3.2 3B with hyperparameters and configurations listed in Table 7. We adopted most settings from previous studies (Hu et al., 2023) but conducted our own learning rate sweep. Following LoRA-XS guidelines, we set $\alpha = r$ for their baseline configuration.

For the GLUE benchmark using RoBERTa-large, you can find the hyperparameter details in Table 8. We mostly adhered to the original configurations from the LoRA paper (Hu et al., 2021) but adjusted the learning rate through a sweep. In line with LoRA-XS settings, we fixed α at 16 for their baseline.

	Mistral-7B / Gemma-2 9B	**Llama-3.2 3B**
Optimizer	AdamW	AdamW
Batch size	1	6
Max. Seq. Len	512	256
Grad Acc. Steps	32	24
Epochs	1	2
Dropout	0	0.05
Learning Rate	1×10^{-4}	2×10^{-3}
LR Scheduler	Cosine	Linear
Warmup Ratio	0.02	0.02

Table 7: Hyperparameter settings for training Mistral-7B and Gemma-2 9B on MetaMathQA, and Llama-3.2 3B on COMMONSENSE170K.

D. Dataset Details

The **MetaMathQA** dataset (Yu et al., 2024) creates mathematical questions by rephrasing existing ones from different viewpoints, without adding new information. We assess this dataset using two benchmarks: **GSM8K** (Cobbe et al., 2021),

	CoLA	RTE	MRPC	SST-2	QNLI	STS-B
Optimizer	AdamW					
Batch size	128					
Max Seq. Len.	256					
Epochs	30	30	30	15	15	30
Dropout	0					
Learning Rate	1×10^{-3}					
LR Scheduler	Linear					
Warmup Ratio	0.06					

Table 8: hyperparameter settings for RoBERTa-large on GLUE.

which consists of grade-school math problems requiring multi-step reasoning, and **MATH** (Hendrycks et al., 2021), which presents difficult, competition-level math problems.

COMMONSENSE170K is a comprehensive dataset that consolidates eight commonsense reasoning datasets (Hu et al., 2023), as described below:

1. **HellaSwag** (Zellers et al., 2019) challenges models to select the most plausible continuation of a given scenario from multiple possible endings.
2. **ARC Easy** (or **ARC-e**) (Clark et al., 2018) includes basic science questions at a grade-school level, offering simpler tasks to assess fundamental reasoning abilities.
3. **PIQA** (Bisk et al., 2020) evaluates physical commonsense reasoning, where models must choose the best action to take in a hypothetical scenario.
4. **SIQA** (Sap et al., 2019) tests social commonsense reasoning by asking models to predict the social consequences of human actions.
5. **WinoGrande** (Sakaguchi et al., 2021) presents sentence completion tasks requiring commonsense reasoning to select the correct binary option.
6. **ARC Challenge** (or **ARC-c**) (Clark et al., 2018) consists of more complex science questions designed to challenge models with sophisticated reasoning, beyond simple co-occurrence patterns.
7. **OBQA** (Mihaylov et al., 2018) features open-book, knowledge-intensive QA tasks that require multi-hop reasoning across multiple information sources.
8. **BoolQ** (Clark et al., 2019) involves answering yes/no questions based on real-world, naturally occurring queries.

The **GLUE Benchmark** is a comprehensive collection of tasks designed to evaluate natural language understanding (NLU) abilities. It included various datasets, including **STS-B** for measuring semantic textual similarity (Cer et al., 2017), **RTE** for recognizing textual entailment, **MRPC** for detecting paraphrases (Dolan & Brockett, 2005), **CoLA** for assessing linguistic acceptability (Warstadt et al., 2019), **SST-2** for sentiment analysis (Socher et al., 2013), and **QNLI** for question-answer inference (Rajpurkar et al., 2018). GLUE's broad scope makes it a standard benchmark for evaluating models like RoBERTa.