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]

[1] Mohamed bin Zayed University of Artificial Intelligence, [2] Georgia Institute of Technology, [3] Massachusetts Institute of Technology

Abstract

Low-rank adapters have become standard 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 learnable $r \times r$ 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 **27-90** times fewer learnable 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.

1 Introduction

While pre-trained foundation models excel at general-purpose capabilities (Bubeck et al., 2023; Hao et al., 2022), adapting them to specific downstream tasks often requires fine-tuning (FT). 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 FT (Liu et al., 2022). At the same time, full FT, while highly effective, is computationally expensive and impractical at scale.

Parameter-efficient fine-tuning (PEFT) has become vital for adapting large language models (LLMs) under computational constraints. Low-rank methods like LoRA (Hu et al., 2021) address this by reducing learnable parameters via low-rank updates, sparking advancements in optimization, initialization, structured matrices, and adaptive rank selection (Zhang et al., 2023; Wang et al., 2024b;a). Low-rank decomposition methods operate on a fundamental premise: FT requires learning only a low-rank update to the pre-trained weights. However, the gradients computed by these methods do not inherently possess this property. For instance, LoRA's gradients need explicit optimization at each step to better approximate the full FT gradient (Wang et al., 2024b). 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).

We analyze these limitations in the context of the architecture of LoRA-XS (Bałazy et al., 2024)—which inserts a learnable $r \times r$ matrix between B and A while keeping other matrices fixed. While exploring solutions inspired by LoRA-based methods, we discover a remarkable property unique to LoRA-XS: through careful initialization of A and B, we can simulate the full FT

[*] Equal contribution. Order decided by coin toss.

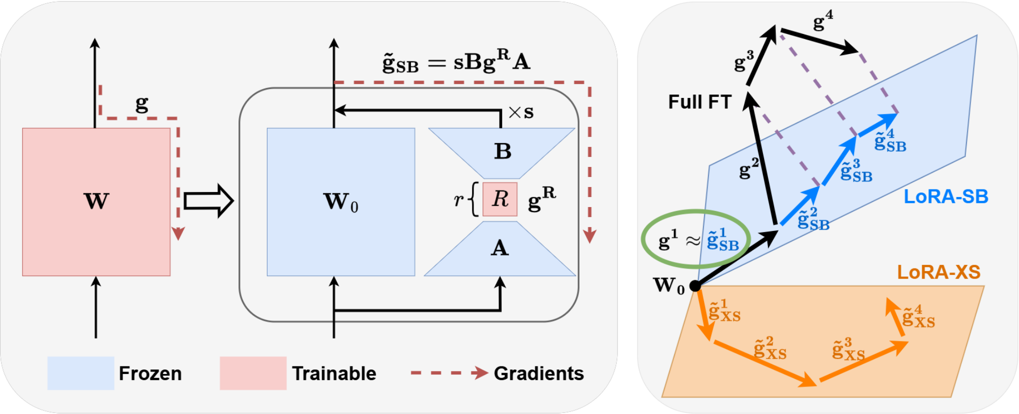

W
g
$\mathbf{W}_0$
$\tilde{\mathbf{g}}_{\mathbf{SB}} = \mathbf{sBg^RA}$
×s
B
r
R
$\mathbf{g^R}$
A
Frozen
Trainable
Gradients
$\mathbf{g^4}$
$\mathbf{g^3}$
Full FT
$\mathbf{g^2}$
$\mathbf{g^1} \approx \tilde{\mathbf{g}}^1_{\mathbf{SB}}$
$\tilde{\mathbf{g}}^2_{\mathbf{SB}}$
$\tilde{\mathbf{g}}^3_{\mathbf{SB}}$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

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

optimization in low rank subspaces through **entire training**, as shown in Figure 1. Our initialization provides optimal scaling for approximating high-rank full FT gradients and eliminates need for scaling the hyperparameter α.

Key Contributions:

- We formalize the limitations of LoRA-XS, showing how its constrained update space leads to suboptimal gradient approximation, initialization sensitivity, and hyperparameter dependence.
- We propose an initialization derived from approximating the first step of full FT, proving it provides optimal low-rank approximation of the initial gradient and preserves update directions throughout.
- We prove that our initialization makes gradient optimization hyperparameter-independent and guarantees convergence, eliminating the need for any tuning of the scaling factor.
- Through extensive experiments across mathematical reasoning, commonsense reasoning, and language understanding tasks, we demonstrate that our method surpasses LoRA's performance while using **27-90x** less learnable parameters, and comprehensively outperforms LoRA-XS.

2 Methodology

2.1 Preliminaries

In standard FT, a pre-trained weight matrix $W \in \mathbb{R}^{m \times n}$ is updated using the update matrix ΔW as: $W = W_0 + \Delta W$, where W_0 is the pre-trained weight. This requires updating mn parameters per layer. LoRA posits that updates lie in a low-dimensional subspace, parameterizing ΔW as: $W = W_0 + sBA$, 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 (α/r) to stabilize training. This reduces the number of parameters from mn to $r(m+n)$. LoRA-XS efficiently parameterizes as: $W = W_0 + sBRA$, where B and A are fixed, and only $R \in \mathbb{R}^{r \times r}$ is trainable, reducing the number of parameters to r^2. We denote the full FT gradient as $g = \frac{\partial L}{\partial W}$, and the LoRA-XS gradient as $g^R_{\text{LoRA-XS}} = \frac{\partial L}{\partial R}$, where L is the loss function.

2.2 Motivation

LoRA-XS (Bałazy et al., 2024) has significantly fewer learnable parameters than LoRA but performs suboptimally. LoRA-XS's architecture causes constraints on the type of updates it can learn. This implies that while ΔW is constrained to be rank $\leq r$, it also needs to have column and row spaces defined by those of B and A, respectively. In contrast, 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 FT 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 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 relies 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 commonly used, including in the original LoRA paper (Hu et al., 2021). However, in LoRA-XS, this results in no learning, as the product BRA remains zero. LoRA-XS uses the most significant subspaces spanned by the columns of pre-trained weights for initialization, inspired by Meng et al. (2024). This initialization is not aligned well with FT because it fails to capture the specific subspaces relevant to the FT task.

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

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

We solve each problem rigorously with proofs in Appendix B. The solutions discussed there 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, derived from approximating the first step of full FT:

$$U, S, V^\top \leftarrow \textbf{SVD}(\Delta W_{avg}) \tag{1}$$

$$A_{init} \leftarrow V[1:r], \quad B_{init} \leftarrow U[1:r], \quad R_{init} \leftarrow \frac{1}{s} S[1:r, 1:r] \tag{2}$$

By the Eckart-Young theorem (Eckart & Young, 1936; Mirsky, 1960), this gives the optimal rank-r approximation of the full FT update. where U, S, V are obtained from truncated SVD of the averaged first update ΔW_{avg}. This initialization leads to several key advantages.

Simplified Gradient Optimization. Our initialization ensures B_{init} and A_{init} form orthonormal bases in $\mathbb{R}^m$ and $\mathbb{R}^n$ respectively, leading to $B^\top B = AA^\top = I$. With fixed B and A matrices being orthonormal, the need for complex matrix inversions during training is eliminated, , as the optimal update step, derived in Equation 2, simplifies to:

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

Optimal Update Approximation. Our initialization guarantees that the first update optimally approximates the full FT weight updates: $sB_{\text{init}}R_{\text{init}}A_{\text{init}} \approx \Delta W_{avg}$. By the Eckart-Young theorem, this gives the optimal rank-r approximation of the initial full FT update.

Hyperparameter Independence. As shown in Theorem 4, when gradient approximation is applied with orthonormal B and A, the hyperparameter s can be set to 1, resulting in:

$$\boxed{g^R = g^R_{\text{LoRA-XS}}} \tag{3}$$

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 3, guaranteeing stable optimization and convergence.

LoRA-SB Advantages over LoRA. Many 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 $(B^\top B)^{-1}$ and $(AA^\top)^{-1}$, 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 $B^\top B$ and $AA^\top$ is required at each optimizer step for accurate gradient approximation.

Algorithm. To optimize GPU memory usage during initialization, we hook into the backward pass of PyTorch and compute the gradients layerwise, immediately discarding the computed gradients (Lv et al., 2024; Wang et al., 2024a). This ensures $O(1)$ memory usage, independent of the number of layers, keeping memory consumption well within manageable limits and **ensuring it does not exceed the memory requirements of subsequent LoRA-SB FT**. For large batch sizes, memory usage can be further optimized through gradient accumulation and quantization. **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.

3 Experiments

Baselines. We compare LoRA-SB against full FT, LoRA, LoRA-XS, rsLoRA, and PiSSA. rsLoRA introduces a rank-scaled stabilization factor ($\alpha/\sqrt{r}$) to enhance stability, while PiSSA updates only the principal components of the pre-trained weight W and freezes the residuals.

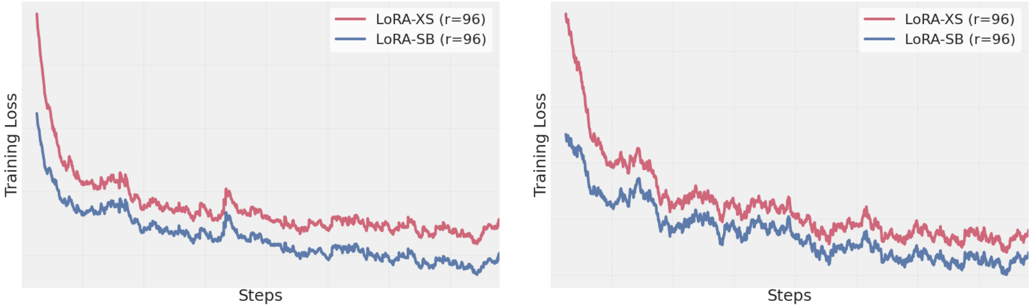
Arithmetic Reasoning. We fine-tune Mistral-7B (Jiang et al., 2023) and Gemma-2 9B (Team et al., 2024) on 50K samples from the MetaMathQA (Yu et al., 2024) dataset and evaluate them on the GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) benchmarks. We apply LoRA modules to the key, value, query, attention output, and all fully connected weight matrices, training with ranks $r = \{32, 64, 96\}$. We present results in Table 1. LoRA-SB significantly outperforms LoRA-XS across all settings. Notably, LoRA-SB outperforms LoRA-based methods ($r = 32$) 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 and converges to a superior final value.

Commonsense Reasoning. We fine-tune Llama-3.2 3B (Dubey et al., 2024) on CommonSense170K, a dataset with 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). LoRA modules are applied to the key, value, query, attention output, and all fully connected weight matrices, training with ranks $r = \{32, 64, 96\}$. We present the results in Table 2. LoRA-SB consistently outperforms LoRA-XS across all settings. In addition, LoRA-SB ($r = 96$) outperforms LoRA-based methods ($r = 32$) with **27x** fewer trainable parameters.

Natural Language Understanding. We present results in Appendix C.6.

Table 1: Accuracy comparison of FT 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**.

Method	**Rank**	**Mistral-7B**			**Gemma-2 9B**		
		# Params	**GSM8K**	**MATH**	**# Params**	**GSM8K**	**MATH**
Full FT	-	7.24 B	63.87	17.65	9.24 B	79.23	38.02
LoRA	32	83.88 M	61.94	15.98	108.04 M	76.19	36.56
rsLoRA	32	83.88 M	62.15	16.24	108.04 M	76.84	36.88
PiSSA	32	83.88 M	62.43	16.52	108.04 M	77.12	37.04
LoRA-XS	32	0.23 M	54.28	13.36	0.30 M	74.07	34.62
LoRA-XS	64	0.92 M	57.08	15.62	1.20 M	75.02	36.46
LoRA-XS	96	2.06 M	58.53	16.42	2.71 M	75.21	36.98
LoRA-SB	32	0.23 M	58.91	15.28	0.30 M	75.44	36.66
LoRA-SB	64	0.92 M	60.73	16.28	1.20 M	76.65	37.14
LoRA-SB	96	2.06 M	**63.38**	**17.44**	2.71 M	**78.40**	**37.70**



(a) Mistral-7B

(b) Gemma-2 9B

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

Table 2: Accuracy comparison of FT methods on Llama-3.2 3B across eight commonsense reasoning datasets. # Params: the number of trainable parameters. Best results among PEFT methods in **bold**.

Method	**Rank**	**# Params**	**Accuracy (↑)**								
			BoolQ	**PIQA**	**SIQA**	**HellaS.**	**WinoG.**	**ARC-e**	**ARC-c**	**OBQA**	**Avg.**
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**	81.30
rsLoRA	32	48.63 M	69.81	85.05	78.92	90.45	82.02	86.71	74.18	81.72	81.11
PiSSA	32	48.63 M	70.12	**85.42**	79.44	90.88	82.68	87.23	74.61	81.79	81.52
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**

4 Conclusion

In this work, we introduced LoRA-SB, which bridges the gap between low-rank PEFT and full FT. This is enabled by our initialization strategy, which approximates the first step of full FT and ensures that the most relevant subspaces for task-specific adaptation are captured. We achieve optimal gradient scaling and preserve update directions throughout training. Our approach ensures hyperparameter independence by approximating the full FT gradient, thereby eliminating instability issues associated with scaling factors. Through extensive experiments, we demonstrate that our method outperforms LoRA while using upto **90x** less parameters, and comprehensively outperforms LoRA-XS. Our work advances 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 integrating LoRA-SB with quantization. We also aim to evaluate its performance on other models, such as Vision Language Models (VLMs) and Vision Transformers (ViTs).

5 ACKNOWLEDGEMENTS

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

REFERENCES

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].

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].

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

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].

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.

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 Related Work

PEFT. 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).

Low-Rank Decomposition Methods. LoRA (Hu et al., 2021) demonstrated that weight updates during FT could be efficiently approximated using low-rank matrices, drastically reducing parameter counts. 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 methods. LoRA-based methods have also been applied in other domains, such as efficient federated FT (Sun et al., 2024; Singhal et al., 2024).

Enhancing LoRA Performance. Recent efforts have focused on optimizing LoRA's performance. PiSSA (Meng et al., 2024) demonstrated improvements 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 FT. 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 more parameter savings. LoRA-XS (Bałazy et al., 2024) achieves this 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 for improvement.

B Solving each problem of LoRA-XS

B.1 Approximation of the full FT gradient

As mentioned, LoRA optimization is mathematically equivalent to full FT 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, which can be simply given by $\mathrm{d}W = -sB(\mathrm{d}R)A$. This implies that updating R with gradient g^R is equivalent to updating W with low rank equivalent gradient $\tilde{g}$ in full FT as described in Definition 1.

Definition 1. *We define the equivalent gradient as:*

$$\tilde{g} = sBg^RA$$

where g^R is the gradient of L with respect to 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 W. To bridge the performance gap between LoRA-XS and full FT, we aim 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 1.

Lemma 1. *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:*

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

Proof. See Appendix C.1. □

We can now formulate our objective to minimize the distance between the equivalent gradient and the full gradient. We do not have access to the full FT gradient g during LoRA-XS based FT. 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 2.

Theorem 2. *The optimal solution for the objective $min_{g^R}||\tilde{g} - g||^2_F$, such that $\tilde{g} = sBg^RA$, is:*

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

Proof. See Appendix C.2. □

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

Theorem 3. *Consider the update for matrix R using the solution derived in Theorem 2:*

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

where $\eta > 0$ is the (sufficiently small) learning rate. This update guarantees a reduction in the loss ΔL, given by:

$$\Delta L = -\eta \langle g^R_{LoRA-XS}, g^R \rangle_F + o(\eta) \leq 0.$$

Proof. See Appendix C.3. □

B.2 Initialization using update approximation

In FT, the primary goal is to update weights to better suit the target task. The initial gradient steps are particularly informative, as they indicate the direction of desired adaptation. We leverage this insight by using the first update step from full FT for initialization.

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 properties. Second, since A and B are fixed, initializing them to span the subspace of early adaptation increases the likelihood of capturing useful updates throughout training. This also ensures that the final update is learnt in the correct subspace, of which we have no apriori information besides the first full FT step. Our method is summarized as: set such initialization that best approximates the first step of full FT. Given a full FT update $\Delta W_{first-step}$, our initialization satisfies:

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

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

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

However, the usage of 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:

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

This 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 this relationship. For instance, we can initialize B and A as US and $V^\top$ and keep R as I/s. This 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.

B.3 Hyperparameter independence

The hyperparameter α is used in LoRA and other decomposition-based method 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 shown in methods like rank stabilization (Kalajdzievski, 2023). However, the full FT gradient g needs no such tuning. We claim that approximating the full FT gradient removes the need for introducing a scaling factor, as shown in Theorem 4.

Theorem 4. *The equivalent gradient $\tilde{g}$ is hyperparameter s independent for $\tilde{g} = sBg^R A$, but not for $\tilde{g} = sBg^R_{LoRA-XS}A$*

Proof. See Appendix C.4. □

The hyperparameter independence of the equivalent gradient eliminates the need for manual gradient scaling. Updates to W depend solely on this gradient (modulo learning rate), making any additional scaling redundant. This can be understood by examining the relationship with the full FT 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. This property is unique to our gradient approximation approach and does not hold for standard LoRA-XS.

B.4 Additional Benefits of LoRA-SB

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.

$$\Delta(sB_{init}R_{init}A_{init}) \approx \gamma \Delta W \tag{8}$$

Thankfully, we don't have to choose between the two. For SGD, we prove that setting B_{init} and A_{init} using Equation 1, results in the first update of LoRA-XS to best approximate the direction of the update of full FT (Theorem 5).

Theorem 5. *If A_{init} and B_{init} are initialized using LoRA-SB for the first step of SGD optimizer, then*

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

Proof. See Appendix C.5. □

C Mathematical Proofs

In all the proofs below, we will use the notations defined in Section 2.

C.1 Proof of Lemma 1

Lemma. *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:*

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

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

$$g := \frac{\partial L}{\partial W} \quad \& \quad g^R_{\text{LoRA-XS}} := \frac{\partial L}{\partial R}. \tag{9}$$

The chain rule gives

$$\frac{\partial L}{\partial R} = \frac{\partial L}{\partial W}\frac{\partial W}{\partial R} \implies \frac{\partial L}{\partial R} = \frac{\partial L}{\partial W}\frac{\partial W}{\partial X}\frac{\partial X}{\partial R} \quad \text{for } X = RA \tag{10}$$

We know that for $W = sBX$:

$$\frac{\partial L}{\partial W}\frac{\partial W}{\partial X} = sB^\top g \implies \frac{\partial L}{\partial R} = sB^\top g \frac{\partial X}{\partial R} \tag{11}$$

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

$$y\frac{\partial X}{\partial R} = yA^\top \implies \frac{\partial L}{\partial R} = yA^\top = sB^\top g A^\top \tag{12}$$

$$\text{Therefore,} \quad \boxed{g^R_{\text{LoRA-XS}} = sB^\top g A^\top} \tag{13}$$

□

C.2 Proof of Theorem 2

Theorem. *The optimal solution for the objective $min_{g^R} ||\tilde{g} - g||^2_F$, such that $\tilde{g} = sBg^R A$, is:*

$$g^R = \frac{1}{s^2}(B^\top B)^{-1} g^R_{LoRA-XS}(AA^\top)^{-1} \tag{14}$$

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

$$\arg\min_{g^R} F = \|sBg^R A - g\|^2_F \tag{15}$$

For differentiable F,

$$\frac{\partial F}{\partial g^R} = 0 \implies 2(\tilde{g} - g) \cdot \frac{\partial \tilde{g}}{\partial g^R} = 0 \implies 2(sBg^R A - g) \cdot \frac{\partial (sBg^R A)}{\partial g^R} = 0 \tag{16}$$

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

$$2sB^\top(sBg^R A - g)A^\top = 0 \implies B^\top(sBg^R A - g)A^\top = 0 \implies B^\top sBg^R AA^\top = B^\top g A^\top \tag{17}$$

From Lemma 1, we get:

$$B^\top g A^\top = g^R_{\text{LoRA-XS}}/s \implies B^\top sBg^R AA^\top = g^R_{\text{LoRA-XS}}/s \implies B^\top Bg^R AA^\top = g^R_{\text{LoRA-XS}}/s^2 \tag{18}$$

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

$$(B^\top B)^{-1}(B^\top B g^R A A^\top)(AA^\top)^{-1} = (B^\top B)^{-1} g^R_{\text{LoRA-XS}}(AA^\top)^{-1}/s^2 \tag{19}$$

$$\text{Therefore,} \quad \boxed{g^R = \frac{1}{s^2}(B^\top B)^{-1} g^R_{\text{LoRA-XS}}(AA^\top)^{-1}} \tag{20}$$

□

C.3 Proof of Theorem 3

Theorem. *Consider the update for matrix R using the solution derived in Theorem 2:*

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

where $\eta > 0$ is the (sufficiently small) learning rate. This update guarantees a reduction in the loss ΔL, given by:

$$\Delta L \coloneqq L(W_0 + sB(R - \eta g^R)A) - L(W_0 + sBRA) = -\eta \langle g^R_{LoRA-XS}, g^R \rangle_F + o(\eta) \leq 0$$

Proof. Assuming that L is differentiable, we use Taylor's theorem and get

$$\begin{aligned}
\Delta L &\coloneqq L(W_0 + sB(R - \eta g^R)A) - L(W_0 + sBRA) \\
&= \left\langle \frac{\partial L}{\partial R}, -\eta g^R \right\rangle_F + o(\eta) \\
&= -\frac{\eta}{s^2}\langle g^R_{\text{LoRA-XS}}, (B^\top B)^{-1} g^R_{\text{LoRA-XS}}(AA^\top)^{-1}\rangle_F + o(\eta),
\end{aligned} \tag{21}$$

where in the last step we also used the definition of $g^R_{\text{LoRA-XS}}$ and the result of Theorem 2. To prove $\Delta L \leq 0$ for small enough η, it is sufficient to show that

$$\langle g^R_{\text{LoRA-XS}}, (B^\top B)^{-1} g^R_{\text{LoRA-XS}}(AA^\top)^{-1}\rangle_F \geq 0. \tag{22}$$

Next, we note that matrices $B^\top B \in \mathbb{R}^{r \times r}$ and $AA^\top \in \mathbb{R}^{r \times r}$ are positive definite since they are positive semi-definite and matrices B and A are full-rank (i.e., with rank r) matrices, which means that $B^\top B$ and $AA^\top$ have non-zero eigenvalues. Therefore, $(B^\top B)^{-1}$ and $(AA^\top)^{-1}$ are also positive definite, implying that there exist matrices U and V such that $(B^\top B)^{-1} = VV^\top$ and $(AA^\top)^{-1} = UU^\top$ (e.g., one can find such matrices using Cholesky decomposition). Then, we have

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

This concludes the proof. □

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

$$\Delta L = -\eta \langle g^R_{\text{LoRA-XS}}, g^R_{\text{LoRA-XS}} \rangle_F + o(\eta) \leq 0. \tag{23}$$

C.4 Proof of Theorem 4

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

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

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 $\tilde{g} = sBg^RA$:

$$g^R = \frac{1}{s^2}(B^\top B)^{-1} g^R_{\text{LoRA-XS}} (AA^\top)^{-1} \implies \tilde{g} = \frac{s}{s^2} B(B^\top B^{-1}) g^R_{\text{LoRA-XS}} (AA^\top)^{-1} A \tag{24}$$

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

$$\tilde{g} = \frac{1}{s} B(B^\top B^{-1}) sB^\top gA^\top (AA^\top)^{-1} A = B(B^\top B^{-1}) B^\top gA^\top (AA^\top)^{-1} A. \tag{25}$$

which is s-independent.

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

$$g^R_{\text{LoRA-XS}} = sB^\top gA^\top \implies \tilde{g} = sB(sB^\top gA^\top)A \implies \tilde{g} = s^2 BB^\top gA^\top A \tag{26}$$

which is not s-independent. □

C.5 Proof of Theorem 5

Theorem. *If A_{init} and B_{init} are initialized using LoRA-SB for the first step of 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:

$$\Delta R = -\eta \nabla_R \mathcal{L}(R) \implies B\Delta RA = -\eta B \nabla_R \mathcal{L}(R) A. \tag{27}$$

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

$$\Delta W = -\eta \nabla_W \mathcal{L}(W_0). \tag{28}$$

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

$$\|B\Delta RA - \eta \nabla \mathcal{L}_W(W_0)\|_F = \eta \|B \nabla_R \mathcal{L}(R) A - \nabla \mathcal{L}_W(W_0)\|_F. \tag{29}$$

We have shown before that:

$$\nabla_R \mathcal{L} = B^\top \nabla_W \mathcal{L} A^\top. \tag{30}$$

The problem now becomes:

$$\min_{A_{\text{init}}, B_{\text{init}}} \|B^\top (B^\top \nabla_W \mathcal{L} A^\top) A - \nabla_W \mathcal{L}\|_F \quad \text{where } \nabla_W \mathcal{L} = USV^\top. \tag{31}$$

Using our initialization, we get:

$$\|BB^\top \nabla_W \mathcal{L} A^\top A - \nabla_W \mathcal{L}\|_F = \|U_{IR} U_{IR}^\top USV^\top V_{IR} V_{IR}^\top - USV^\top\|_F. \tag{32}$$

Moreover, we also have

$$U_{IR} U_{IR}^\top USV^\top V_{IR} V_{IR}^\top = \sum_{i=1}^{r} \sigma_i u_i v_i^\top. \tag{33}$$

The rank of W' such that

$$W' = U_{IR} U_{IR}^\top USV^\top V_{IR} V_{IR}^\top \tag{34}$$

is $\leq r$, since the corresponding ranks of B_{init} and A_{init} is r. Using the Eckart-Young Theorem, we find the optimal low-rank solution as:

$$W'^{*} = \underset{\text{rank}(W')=r}{\arg\min} \|W' - \nabla_W \mathcal{L}\|_F = \sum_{i=1}^{r} \sigma_i u_i v_i^\top. \tag{35}$$

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

C.6 Natural Language Understanding

We fine-tune RoBERTa-large (Liu et al., 2019) on GLUE, a popular language understanding classification benchmark that contains several datasets. 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, with ranks $r = \{8, 16, 24\}$. The results are shown in Table 3. LoRA-SB consistently outperforms LoRA-XS across all configurations. Additionally, LoRA-SB ($r = 24$) outperforms LoRA-based methods ($r = 8$) with **39x** lesser trainable parameters.

Table 3: Comparison of FT 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.

Method	Rank	# Params	CoLA	RTE	MRPC	STS-B	QNLI	SST-2	All
			Mcc ↑	Acc ↑	Acc ↑	Corr ↑	Acc ↑	Acc ↑	Avg. ↑
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
rsLoRA	8	2162.69 K	67.87	82.84	89.97	91.30	93.29	95.87	86.85
PiSSA	8	2162.69 K	68.22	**83.14**	90.10	91.59	93.55	96.03	87.10
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	93.69	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**

D Analysis

D.1 Optimal Initialization is Important!

To isolate the impact of initialization, we take truncated SVD on various matrices, including Kaiming initialization (He et al., 2015) and ΔW_{avg} with varying levels of 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**

D.2 Optimal Gradient Approximation is Important!

We aim to examine the effect of optimal gradient approximation. 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:

$$U, S, V^T \leftarrow \textbf{SVD}(\Delta W_{avg}) \tag{36}$$

$$B_{\text{init}} \leftarrow U[1:r]S[1:r, 1:r] \tag{37}$$

$$A_{\text{init}} \leftarrow V[1:r] \tag{38}$$

$$R_{\text{init}} \leftarrow I \tag{39}$$

This construction ensures that $B_{\text{init}} R_{\text{init}} A_{\text{init}} \approx \Delta W_{avg}$, but only $AA^T = I$, while $B^T B \neq I$. The setup is suboptimal for gradient approximation since we do not explicity use the closed-form solution derived in Theorem 2. 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 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 GSM8K and MATH, with scores of 63.38 and 17.44 compared to 55.87 and 12.74, respectively.

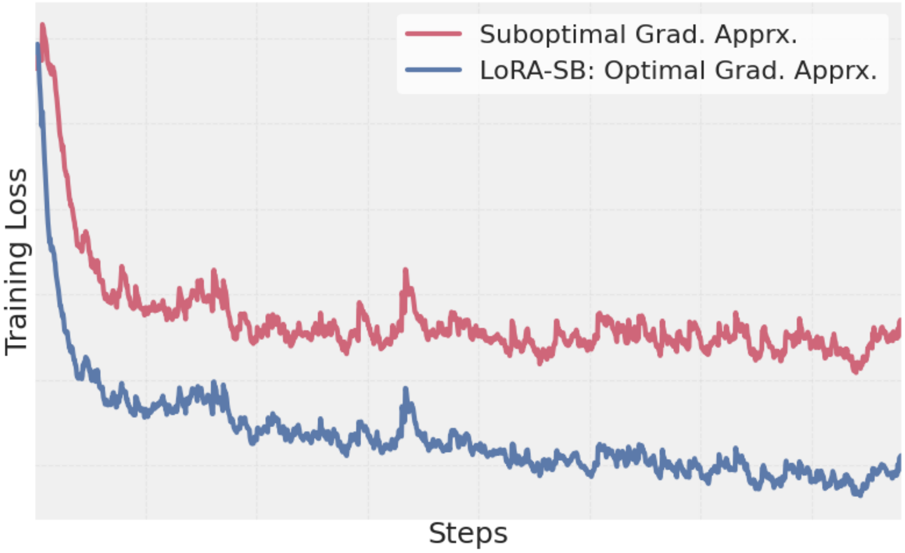


Figure 3: Training loss for Mistral-7B on MetaMathQA, highlighting the impact of optimal gradient approximation.

D.3 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.

D.4 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. This comparison is presented in Table 6, showing that the additional overhead of LoRA-SB is negligible.

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

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

E Experiment Details

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 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 use PyTorch (Paszke et al., 2019) and the HuggingFace Transformers library (Wolf et al., 2020) for our implementations. We run all experiments on a **single NVIDIA A6000 GPU** and report results as the average of three random seeds. We trained all models using the AdamW optimizer (Loshchilov & Hutter, 2019). To save memory, we initialize base models in **`torch.bfloat16`** precision. Appendix F 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. The samples are randomly selected from the training set in each run.

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.

For all tasks, we followed the baseline configurations provided in the PiSSA (Meng et al., 2024) and rsLoRA (Kalajdzievski, 2023) papers for our comparisons.

	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.

	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.

F 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), which consists of grade-school math problems requiring multi-step reasoning, and **MATH** (Hendrycks et al., 2021), which presents difficult, competition-level math problems. Evaluation focuses solely on the final numeric answer.

COMMONSENSE170K is a comprehensive dataset that consolidates eight commonsense reasoning datasets (Hu et al., 2023). 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). The individual datasets used are 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.