SOS-LoRA: Static Orthogonal-Subspace Low-Rank Adaptation with Fixed Multi-Scale Scaling

Yupeng Chang[1] **Yuan Wu**[1*] **Yi Chang**[1,2,3]
[1]School of Artificial Intelligence, Jilin University
[2]Engineering Research Center of Knowledge-Driven Human-Machine Intelligence, MOE, China
[3]International Center of Future Science, Jilin University
changyp23@mails.jlu.edu.cn, {yuanwu, yichang}@jlu.edu.cn

Abstract

Low-Rank Adaptation (LoRA) is a widely used parameter-efficient fine-tuning (PEFT) method for large language models. Under a fixed rank budget, LoRA parameterizes each adapted weight through a single low-dimensional input-side pathway, which may couple heterogeneous behaviors through shared input directions and induce interference during optimization. We propose **Static Orthogonal Subspace LoRA** (SOS-LoRA), a drop-in extension that reparameterizes a rank-r_{tot} update as a sum of K *static* (always-on, non-routed) low-rank experts. SOS-LoRA (i) decomposes the total rank across experts, (ii) applies a *fixed* multi-scale scaling scheme to encourage scale-separated optimization dynamics, and (iii) promotes diverse input-side directions via cross-expert orthogonal initialization and a lightweight regularizer. SOS-LoRA remains fully mergeable, adding no inference-time parameters or latency after merging. Experiments on reasoning and knowledge-intensive benchmarks (Llama 2/3), encoder-based NLU (GLUE), and math reasoning (GSM8K/MATH) show consistent gains over matched-budget LoRA baselines and recent variants. Code is available at https://github.com/llm172/sos-lora.

1 Introduction

Parameter-Efficient Fine-Tuning (PEFT) is widely used to adapt large language models (LLMs) (Touvron et al., 2023; Chang et al., 2024; Han et al., 2024) when full fine-tuning is computationally and operationally expensive. Among PEFT strategies, Low-Rank Adaptation (LoRA) (Hu et al., 2022) is a standard choice due to its simplicity, strong empirical performance, and deployment-friendly mergeability. LoRA is motivated by the observation that task-specific weight updates often exhibit low effective dimensionality, consistent with evidence on low intrinsic dimension in fine-tuning (Aghajanyan et al., 2020). Concretely, LoRA freezes pretrained weights W_0 and learns a rank-r_{tot} update $\Delta W = \frac{\alpha}{r_{\text{tot}}} AB$, where $A \in \mathbb{R}^{m \times r_{\text{tot}}}$ and $B \in \mathbb{R}^{r_{\text{tot}} \times n}$, yielding the adapted weight $W_0 + \Delta W$. Since the update is additive and linear, it can be merged into W_0 after training, preserving the backbone architecture and inference efficiency (Hu et al., 2022).

Despite its success, standard LoRA assigns each adapted matrix a *single* low-rank update pathway. This raises a natural question: under a fixed rank budget, can one shared low-dimensional pathway adequately support the diverse behaviors required by modern LLMs? In the standard factorization, the adaptation signal depends on the input only through a single projection XA, i.e., a single set of input-side directions shared by all adaptation effects. Consequently, heterogeneous behaviors may be forced to share and compete for the same input-side directions under a fixed rank budget, which can induce interference and optimization coupling. Increasing the rank can partially alleviate this issue, but it also increases trainable parameters and still provides no explicit mechanism for structured adaptation under a fixed budget.

In this work, we mitigate this coupling by *structuring* the low-rank update into multiple expert components without increasing the total rank budget. We propose **Static Orthogonal Subspace LoRA** (**SOS-LoRA**), a drop-in replacement for standard LoRA that reparameterizes a rank-r_{tot} update as a sum of K *static* (always-on, non-routed) low-rank experts. Each expert uses its own input-side directions, is assigned a fixed scale, and is explicitly encouraged to remain directionally distinct from other experts. Importantly, SOS-LoRA does not enlarge the hypothesis class beyond rank-r_{tot} updates; rather, it introduces an optimization-oriented inductive bias that promotes directionally diverse adaptation under the same low-rank budget.

SOS-LoRA instantiates this inductive bias through three components. First, a *parallel ex-*

*Corresponding author

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3993–4005

pert decomposition splits the total rank budget into K experts of rank $r' = r_{\text{tot}}/K$, matching the parameter budget of standard LoRA at the same r_{tot}. Second, a *fixed multi-scale scaling* scheme assigns each expert a static scale spanning small-to-large magnitudes; this scales the raw backpropagated gradients entering each expert and encourages scale-separated learning dynamics. Third, *orthogonal input-side diversification* combines cross-expert orthogonal initialization (with zero initial update) and an orthogonality regularizer that penalizes cross-expert correlations in input-side directions.

Crucially, SOS-LoRA preserves LoRA's key deployment property: after training, all expert updates can be merged into the pretrained weights as a single effective update matrix. Deployment therefore uses the same backbone architecture and per-token FLOPs as the base model, incurring no additional inference latency after merging (Hu et al., 2022).

Our contributions are threefold:

- We identify a key limitation of standard LoRA under a fixed rank budget: a single shared input-side projection can couple heterogeneous behaviors through a common low-dimensional update pathway, which can induce interference and optimization coupling.
- We propose **SOS-LoRA**, which reparameterizes a rank-r_{tot} LoRA update as a sum of K *static* low-rank experts with fixed multi-scale scaling and explicit cross-expert diversification of input-side directions, while preserving LoRA's mergeability and inference efficiency.
- We empirically evaluate SOS-LoRA on reasoning and knowledge-intensive benchmarks, encoder-based NLU tasks (GLUE), and math reasoning (GSM8K/MATH), showing consistent improvements over matched-budget LoRA baselines and recent LoRA variants under a unified training and evaluation pipeline.

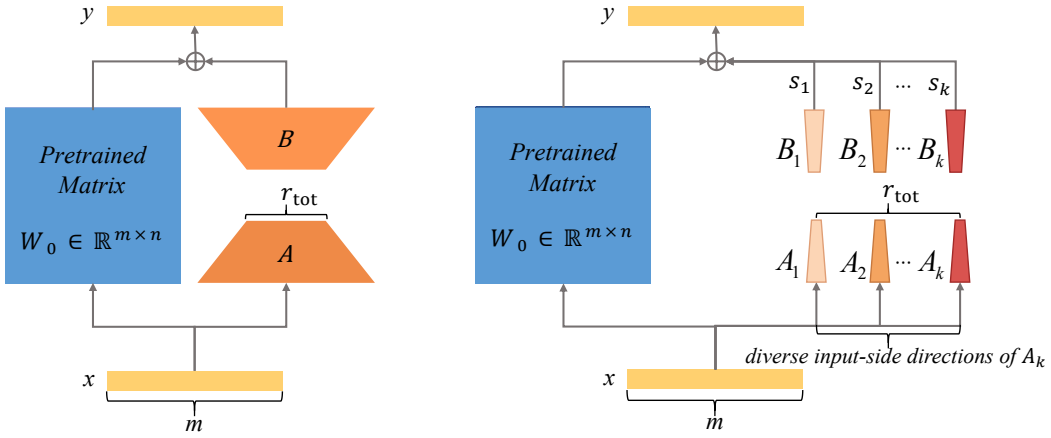
2 Related Work

Parameter-Efficient Fine-Tuning (PEFT) adapts large pretrained models by updating only a small subset of parameters (Han et al., 2024). Representative approaches include adding lightweight modules such as Adapters (Houlsby et al., 2019), prompt-based tuning that optimizes continuous prompts or prefixes (Lester et al., 2021; Li and Liang, 2021), and low-dimensional reparameterizations of weight updates. Among these, Low-Rank Adaptation (LoRA) (Hu et al., 2022) and its subsequent extensions (Chang et al., 2026, 2025) freeze pretrained weights and learn efficient low-rank updates, making LoRA-based methods a widely used PEFT family.

A growing body of work extends LoRA along multiple axes. Optimization- and training-focused variants aim to improve convergence or update quality (e.g., LoRA+ (Hayou et al., 2024), LoRA-GA (Wang et al., 2024a), LoRA-Pro (Wang et al., 2024b)), while orthogonality-based methods encourage less interfering low-rank updates (e.g., O-LoRA (Wang et al., 2023), primarily studied in continual learning). Other directions explore multiple LoRA experts through token-dependent routing (e.g., MixLoRA (Li et al., 2024)), combine multiple *trained* LoRAs via learnable gating (e.g., MoLE (Wu et al., 2024)), or introduce multi-scale designs across layers (e.g., MSPLoRA (Zhao et al., 2025)). In contrast, **SOS-LoRA** preserves LoRA's mergeability and a fixed total rank budget, but reparameterizes the update *within each adapted weight* as a sum of K *static* (always-on, non-routed) low-rank experts. It further employs a *fixed* multi-scale scaling scheme and explicitly promotes *cross-expert diversity of input-side directions* via an orthogonality regularizer, while remaining expressively equivalent to standard LoRA under the same total rank.

3 Methodology

We propose **Static Orthogonal Subspace LoRA (SOS-LoRA)**, a parameter-efficient fine-tuning (PEFT) method that aims to improve *effective* adaptation under a fixed low-rank budget. SOS-LoRA does so by: (i) decomposing a single rank-r_{tot} update into multiple *static* (always-on, non-routed) low-rank experts; (ii) using a *pre-specified* multi-scale scaling schedule (optionally with mild *input-independent* calibration) to promote stable, scale-separated optimization dynamics; and (iii) explicitly encouraging *cross-expert diversity of input-side directions* through initialization and a lightweight regularizer. Like standard LoRA (Hu et al., 2022), SOS-LoRA is *mergeable* into linear layers: after fine-tuning, the learned low-rank updates can be merged into the base weights, yielding zero additional inference-time parameters and latency after merging. Figure 1 provides an overview of the stan-



(a) LoRA (b) SOS-LoRA (ours)

Figure 1: Overview of Static Orthogonal Subspace LoRA (SOS-LoRA) with row-vector convention $y = xW_0$, $W_0 \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^{1\times m}$, and $y \in \mathbb{R}^{1\times n}$. (a) LoRA freezes W_0 and learns a single rank-r_{tot} update $\Delta W = \frac{\alpha}{r_{\text{tot}}} AB$ with $A \in \mathbb{R}^{m\times r_{\text{tot}}}$ and $B \in \mathbb{R}^{r_{\text{tot}}\times n}$, yielding $y = x(W_0 + \Delta W)$; under this parameterization, the adapter depends on the input only through the shared projection xA. (b) SOS-LoRA reparameterizes the same total rank budget as K *static* (always-on, non-routed) rank-r' experts $\{(A_k, B_k)\}_{k=1}^{K}$ with $r' = r_{\text{tot}}/K$, combined by fixed multi-scale scales $\{s_k\}$ and encouraged to use diverse input-side directions via an orthogonality regularizer on $\{A_k\}$. The merged update is $\Delta W = \sum_{k=1}^{K} s_k A_k B_k$, which can be absorbed into W_0 with no additional inference-time parameters or latency after merging.

dard LoRA parameterization and the SOS-LoRA design.

3.1 Problem Formulation: Input-Side Coupling Under a Fixed Rank Budget

Let $W_0 \in \mathbb{R}^{m\times n}$ be a pre-trained weight matrix in a Transformer linear layer. Let $X \in \mathbb{R}^{N\times m}$ denote the input activations for N token instances (flattened across batch and sequence), and let $Y \in \mathbb{R}^{N\times n}$ denote the corresponding outputs. Standard Low-Rank Adaptation (LoRA) (Hu et al., 2022) freezes W_0 and parameterizes an additive update ΔW with rank at most r_{tot}:

$$Y = X(W_0 + \Delta W) = X\left(W_0 + \frac{\alpha}{r_{\text{tot}}} AB\right), \quad (1)$$

where $A \in \mathbb{R}^{m\times r_{\text{tot}}}$, $B \in \mathbb{R}^{r_{\text{tot}}\times n}$, and α is a scaling scalar.

Shared input-side pathway. Eq. (1) implies that the LoRA path depends on the input only through a single low-dimensional projection. Writing $Z = XA \in \mathbb{R}^{N\times r_{\text{tot}}}$, we have $XAB = ZB$, hence the LoRA contribution depends on X only via Z. In particular, for any X_1, X_2 such that $X_1 A = X_2 A$, the LoRA path yields identical outputs $X_1 AB = X_2 AB$. Equivalently, for a single token row $x \in \mathbb{R}^{1\times m}$, the LoRA path depends on x only through xA (or $A^\top x^\top$). Under a fixed rank budget, this means that all input-sensitive adaptation effects are mediated through the same shared rank-r_{tot} input projection.

Optimization coupling under a fixed rank budget. Increasing r_{tot} increases the maximum rank of ΔW but also increases the number of trainable parameters. Under a *fixed* r_{tot}, a single factorization AB can couple heterogeneous adaptation signals by forcing them to share and compete for the same input-side directions in A. SOS-LoRA introduces a structured parameterization and explicit cross-expert direction regularization to bias optimization toward more diverse input directions, while preserving the same overall rank budget.

3.2 SOS-LoRA: Architecture Design

SOS-LoRA replaces a single rank-r_{tot} adapter with K parallel rank-r' experts, where $r' = r_{\text{tot}}/K$ (assuming K divides r_{tot}). The design has three components: (i) parallel expert decomposition, (ii) multi-scale scaling, and (iii) input-side diversification via orthogonal initialization and regularization.

Expressivity equivalence (hypothesis class). SOS-LoRA does *not* enlarge the set of representable rank-r_{tot} updates compared to standard LoRA; instead, it provides an optimization-oriented inductive bias through scale-separated optimization and cross-expert direction diversity. Formally:

Proposition 1 (Expressivity equivalence)**.** *Let* $\{(A_k, B_k)\}_{k=1}^{K}$ *with* $A_k \in \mathbb{R}^{m\times r'}$ *and* $B_k \in \mathbb{R}^{r'\times n}$*, and let* s_k *be scalars. Define the column concatenation* $A_{\text{cat}} = [A_1, \ldots, A_K] \in \mathbb{R}^{m\times r_{tot}}$ *and the row concatenation* $B_{\text{cat}} = [s_1 B_1; \ldots; s_K B_K] \in \mathbb{R}^{r_{tot}\times n}$*. Then* $\sum_{k=1}^{K} s_k A_k B_k = A_{\text{cat}} B_{\text{cat}}$*, so SOS-LoRA represents the same set of updates with rank at most* r_{tot} *as a single LoRA adapter (up to rank deficiency).*

3.2.1 Parallel Experts and Multi-Scale Scaling

We denote expert parameters by $\{(A_k, B_k)\}_{k=1}^{K}$, where $A_k \in \mathbb{R}^{m\times r'}$ and $B_k \in \mathbb{R}^{r'\times n}$. The SOS-LoRA forward pass is

$$Y_{\text{SOS}} = XW_0 + \sum_{k=1}^{K} s_k\, X A_k B_k, \tag{2}$$

where $s_k > 0$ controls the contribution of expert k. Here, "static" means that (i) all experts are active for all tokens, (ii) there is no token-dependent routing, and (iii) the decomposition introduces no conditional computation.

Fixed multi-scale base scaling. We initialize expert scales as

$$\begin{aligned} s_k &= \frac{\alpha}{r_{\text{tot}}} \cdot \frac{\gamma_k}{\bar{\gamma}}, \\ \gamma_k &= 1 + \frac{k-1}{\max(1, K-1)}(\gamma_{\max} - 1), \\ \bar{\gamma} &= \frac{1}{K}\sum_{k=1}^{K} \gamma_k, \end{aligned} \tag{3}$$

so that the *average* scaling matches standard LoRA, $\frac{1}{K}\sum_{k=1}^{K} s_k = \alpha / r_{\text{tot}}$, while experts receive linearly spaced relative scales $\gamma_k \in [1, \gamma_{\max}]$. Unless stated otherwise, we use $\gamma_{\max} = 2.5$.

Scale-separated gradient signals. Let $\Delta = \partial \mathcal{L} / \partial Y_{\text{SOS}} \in \mathbb{R}^{N\times n}$ denote the upstream gradient for loss $\mathcal{L}$. From Eq. (2), the per-expert gradients satisfy

$$\frac{\partial \mathcal{L}}{\partial B_k} = s_k\, (X A_k)^\top \Delta, \qquad \frac{\partial \mathcal{L}}{\partial A_k} = s_k\, X^\top (\Delta B_k^\top), \tag{4}$$

so s_k linearly scales the raw backpropagated gradients entering expert k. With adaptive optimizers, the resulting parameter *updates* are not strictly linear in s_k due to normalization effects; nevertheless, fixed multi-scale scaling provides a simple and stable mechanism to bias experts toward different update magnitudes.

Optional input-independent calibration (training-time only). In practice, we optionally allow a mild *input-independent* reweighting across experts, initialized from the multi-scale scaling in Eq. (3). This preserves the always-on structure (no routing), can be absorbed into the low-rank factors (hence remains mergeable), and is used purely as a stabilization mechanism during training. In our experiments, unless stated otherwise, we adopt a LoRA+ optimizer setting that uses different effective learning rates for the LoRA factors (Hayou et al., 2024); this is an optimizer-side choice and orthogonal to the SOS-LoRA parameterization.

3.2.2 Orthogonal Input-Side Diversification

To reduce redundancy and training-time interference among experts, we encourage diversity of input-side directions across $\{A_k\}$ through initialization and a lightweight regularizer.

Cross-expert orthogonal initialization with zero initial update. Assuming $r_{\text{tot}} = Kr' \le m$, we initialize experts to span diverse input-side directions while keeping the initial update exactly zero. Concretely, we sample $G \in \mathbb{R}^{m\times(Kr')}$ with i.i.d. Gaussian entries, compute a thin QR factorization $G = QR$ with $Q^\top Q = I_{Kr'}$, and partition Q into K blocks $Q = [Q_1, \ldots, Q_K]$ with $Q_k \in \mathbb{R}^{m\times r'}$. We set $A_k = Q_k$ and initialize $B_k = 0$ for all k. This yields $\Delta W = \sum_{k=1}^{K} s_k A_k B_k = 0$ at initialization, preserving the backbone behavior at the start of fine-tuning.* Under this scheme, B_k receives gradients immediately (Eq. (4)), while $\partial \mathcal{L}/\partial A_k = 0$ at initialization since $B_k = 0$. Thus, A_k starts updating only after B_k becomes non-zero, which stabilizes early training while starting from cross-expert-diverse input directions.

Cross-expert input-direction regularization. We regularize the *input-side directions* by penalizing cross-expert correlations between the column spaces of $\{A_k\}$. Let $\tilde{A}_k \in \mathbb{R}^{m\times r'}$ denote the column-wise ℓ_2-normalized version of A_k, i.e., $\tilde{a} = a/(\|a\|_2 + \varepsilon)$ for each column a and a small $\varepsilon > 0$. We define

$$\mathcal{L}_{\text{orth}} = \frac{\lambda}{r'^2} \sum_{1\le k<\ell\le K} \left\| \tilde{A}_k^\top \tilde{A}_\ell \right\|_F^2, \tag{5}$$

*If $r_{\text{tot}} > m$, one can fall back to per-expert orthogonalization (orthogonalizing each A_k independently), which preserves within-expert orthogonality but cannot guarantee global cross-expert orthogonality.

which equals the sum of squared cosine similarities between all pairs of normalized columns across different experts. This directly discourages experts from collapsing to the same input-side directions while leaving within-expert structure unconstrained. In implementation, λ can be scheduled (e.g., delayed ramp-up), and Eq. (5) can be implemented either as an explicit loss term or via an equivalent gradient form without changing the underlying regularization objective.

4 Experiments

Models and Datasets. We consider decoder-only LLMs (Llama 2-7B (AI@Meta, 2023), Llama 3-8B (AI@Meta, 2024)) and an encoder-only model (RoBERTa-base (Liu et al., 2019)). Our benchmarks cover three capability groups: (i) reasoning and knowledge-intensive QA: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SocialIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-E/ARC-C (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018); (ii) natural language understanding: GLUE (Wang et al., 2018); and (iii) math reasoning: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021).

Compared Methods. We compare SOS-LoRA with full fine-tuning (Full FT) as a reference and standard LoRA (Hu et al., 2022) as the primary PEFT baseline. We additionally include representative LoRA variants that appear in the main results tables: DoRA (Liu et al., 2024) (decomposing weights into magnitude and direction, with low-rank adaptation on the directional component), AdaLoRA (Zhang et al., 2023) (adaptive rank allocation), DyLoRA (Valipour et al., 2022) (supporting a range of effective ranks), MELoRA (Ren et al., 2024) (a mini-ensemble of low-rank adapters), PiSSA (Meng et al., 2024) (SVD-based initialization), Delta-LoRA (Zi et al., 2023) (propagating updates via deltas of successive low-rank products), MixLoRA (Li et al., 2024) (a routing-based sparse mixture of LoRA experts inserted into feed-forward blocks), and MSP-LoRA (Zhao et al., 2025) (a multi-scale pyramid LoRA with shared and layer-specific components). For the cost analysis in Table 3, we report LoRA and Full FT as references; additional PEFT baselines are included where noted.

Implementation Details. All experiments are implemented in PyTorch with Hugging Face Transformers. Unless otherwise noted, SOS-LoRA is applied to all linear projections: for decoder-only LLMs (the Llama family), we adapt the attention projections (`q,k,v,o`) and MLP projections (`up,gate,down`); for RoBERTa-base, we adapt the self-attention and feed-forward linear layers. Unless otherwise specified, we use a fixed total rank $r_{\text{tot}} = 8$ per adapted matrix, decomposed into $K = 4$ static experts ($r' = 2$), with LoRA dropout rate 0.05. We use fixed multi-scale scaling with $\gamma_{\max} = 2.5$ (Eq. 3) and orthogonal initialization with cross-expert regularizer weight $\lambda = 0.01$ (Eq. 5). We optimize with AdamW (Loshchilov and Hutter, 2019) and a linear learning-rate schedule with warmup ratio 0.1. Within each benchmark, we match the training data and training budget across methods. For RoBERTa-base on GLUE, we train for 3 epochs with learning rate 2×10^{-4} and batch size 32 (max sequence length 512). Unless otherwise stated, Llama experiments use 3 epochs with learning rate 3×10^{-5}, per-device batch size 4 with 8 gradient accumulation steps (effective batch size 32), and max sequence length 2048. For analyses that vary the rank budget or expert count (Figs. 4 and 5), we sweep r_{tot} and/or K while keeping all other settings fixed. For the cost analysis in Table 3, we fine-tune Llama 2-7B on the first 100k MetaMathQA samples (Yu et al., 2023) using DeepSpeed (Rasley et al., 2020) ZeRO-2 on 2× NVIDIA A40 (48GB), and report peak *reserved* VRAM summed over the two devices together with end-to-end wall-clock time. The Full FT accuracies follow the protocol described in the table note. Unless stated otherwise, reported results are averaged over three independent runs with different random seeds; we report mean ± standard deviation where applicable.

We evaluate SOS-LoRA on eight reasoning and knowledge-intensive benchmarks. As shown in Table 1, SOS-LoRA achieves the highest average among the compared PEFT methods on both backbones. On Llama 2-7B, SOS-LoRA reaches 85.5 on average, improving over PiSSA (84.5) by 1.0 point and standard LoRA (77.6) by 7.9 points. SOS-LoRA also achieves the best result on all eight benchmarks in this setting, including PIQA (87.5 vs. 87.0 for PiSSA), and attains the best overall average. We also evaluate MixLoRA, a routing-based sparse MoE baseline, under the same pipeline: while it remains competitive on sev-

Table 1: Main results on eight reasoning and knowledge-intensive benchmarks for the Llama family. We compare SOS-LoRA with strong PEFT baselines under a unified evaluation pipeline. Within each model block, the best and second-best PEFT results are marked in bold and underlined, respectively. We additionally report ChatGPT as a reference (see Appendix for evaluation details).

Model	Method	BoolQ	PIQA	SIQA	Hella Swag	Wino Grande	ARC-e	ARC-c	OBQA	Avg.
ChatGPT		73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
Llama 2-7B	LoRA	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
	DoRA	71.8	83.7	76.0	89.1	82.6	83.7	68.2	82.4	79.7
	MixLoRA	72.0	82.8	77.3	92.9	75.7	77.4	57.5	81.1	77.1
	PiSSA	<u>75.0</u>	<u>87.0</u>	<u>81.6</u>	<u>95.0</u>	<u>86.5</u>	<u>88.5</u>	<u>75.9</u>	<u>86.4</u>	<u>84.5</u>
	SOS-LoRA	**75.7**	**87.5**	**83.2**	**95.8**	**87.8**	**89.2**	**76.9**	**88.1**	**85.5**
Llama 3-8B	LoRA	70.8	85.2	79.9	91.7	84.3	84.2	71.2	79.0	80.8
	DoRA	74.6	89.3	79.9	95.5	85.6	90.5	80.4	85.8	85.2
	MixLoRA	74.6	87.2	78.1	93.0	81.4	86.1	79.3	84.0	83.0
	PiSSA	**77.2**	<u>90.0</u>	<u>82.9</u>	<u>96.6</u>	<u>88.4</u>	**93.6**	<u>82.4</u>	<u>87.4</u>	<u>87.3</u>
	SOS-LoRA	<u>77.1</u>	**90.5**	**83.3**	**97.2**	**89.6**	<u>93.4</u>	**83.9**	**89.2**	**88.0**

eral benchmarks, its overall average is below both standard LoRA and SOS-LoRA (77.1 vs. 77.6 and 85.5, respectively). The trend persists on Llama 3-8B: SOS-LoRA attains an average of 88.0, surpassing PiSSA (87.3) by 0.7 and DoRA (85.2) by 2.8 points, while MixLoRA achieves 83.0. Overall, these results are consistent with SOS-LoRA's inductive bias: under a fixed rank budget, decomposing the update into static experts with pre-specified multi-scale scaling and encouraging cross-expert input-side direction diversity provides more effective adaptation than a single low-rank update.

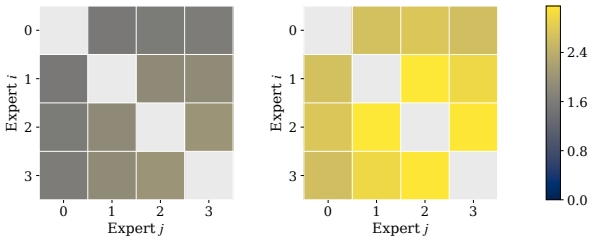


Figure 2: Orthogonal regularization reduces cross-expert similarity in input-side directions. Left: SOS-LoRA with the full method. Right: SOS-LoRA without $\mathcal{L}_{\text{orth}}$. Each entry is $\frac{1}{r'^2}\|\tilde{A}_k^\top \tilde{A}_\ell\|_F^2$, i.e., the mean squared cosine similarity between ℓ_2-normalized input-side directions across experts (K=4), averaged over 224 adapted modules and scaled by 10^6. Removing $\mathcal{L}_{\text{orth}}$ increases off-diagonal similarity.

4.1 Generalization to Encoder Architectures and NLU Tasks

To assess generalization to encoder architectures and NLU tasks, we evaluate SOS-LoRA on GLUE with RoBERTa-base (Liu et al., 2019). As shown in Table 2, SOS-LoRA achieves the highest average score (88.3) among the compared PEFT methods, improving over the strongest PEFT baselines under our evaluation pipeline (Delta-LoRA/MELoRA, 87.5) by 0.8 points, and also yielding a higher average than full fine-tuning in this setting (88.3 vs. 86.8). We additionally report MSP-LoRA (Zhao et al., 2025) as a recent multi-scale LoRA baseline, which attains 86.6 on average in our evaluation. SOS-LoRA achieves the best result on seven of the eight tasks, while Full FT is best on QQP, with consistent gains on paraphrase identification (MRPC), semantic similarity regression (STS-B), and natural language inference (RTE/QNLI/MNLI). Overall, these results indicate that decomposing a fixed low-rank budget into diversified static experts remains effective in encoder-based NLU settings.

4.2 Cross-Expert Input-Side Direction Diversity Analysis

To assess whether SOS-LoRA learns diverse input-side directions, we analyze the A factors after training. For each adapted linear module (224 in total), we compute a $K \times K$ cross-expert similarity matrix using the same squared-cosine measure as in Eq. 5, namely $\frac{1}{r'^2}\|\tilde{A}_k^\top \tilde{A}_\ell\|_F^2$, where $\tilde{A}_k$ denotes column-

Table 2: Performance comparison on the GLUE benchmark using RoBERTa-base. We evaluate SOS-LoRA against full fine-tuning (Full FT) and PEFT baselines under our unified training and evaluation pipeline. For each task, the best result is in bold and the second-best is underlined.

Method	MRPC	RTE	CoLA	STS-B	SST-2	QQP	QNLI	MNLI	Avg.
Full FT	88.2	84.1	64.6	90.6	94.3	**92.0**	92.7	<u>87.5</u>	86.8
LoRA	89.9	85.9	62.4	91.4	94.4	90.8	92.6	86.9	86.8
DyLoRA	89.5	84.5	61.1	91.1	94.3	90.2	92.2	86.3	86.2
AdaLoRA	90.2	85.2	61.6	91.2	94.5	90.1	93.1	87.3	86.7
Delta-LoRA	90.2	<u>87.0</u>	63.8	91.6	95.1	90.9	93.1	<u>87.5</u>	<u>87.5</u>
MELoRA	<u>90.9</u>	86.6	64.1	<u>91.9</u>	<u>95.4</u>	90.8	<u>93.2</u>	87.2	<u>87.5</u>
MSP-LoRA	90.1	80.8	<u>65.3</u>	91.0	94.4	91.2	92.7	87.2	86.6
SOS-LoRA	**91.7**	**88.1**	**65.6**	**92.4**	**95.7**	<u>91.5</u>	**93.9**	**87.6**	**88.3**

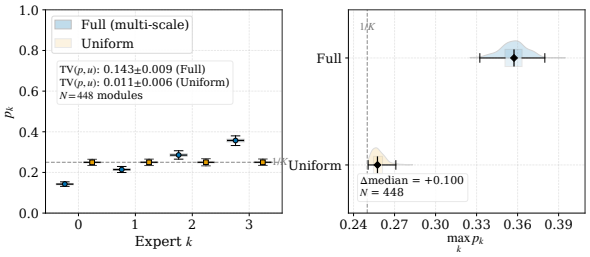


Figure 3: Multi-scale expert scaling leads to more separated per-expert update magnitudes. Left: per-expert relative update magnitude $p_k = \|\Delta W_k\|_F / \sum_{j=1}^{K} \|\Delta W_j\|_F$ across modules (K=4). Right: distribution of the top-1 share $\max_k p_k$, where multi-scale scaling yields a more concentrated contribution profile.

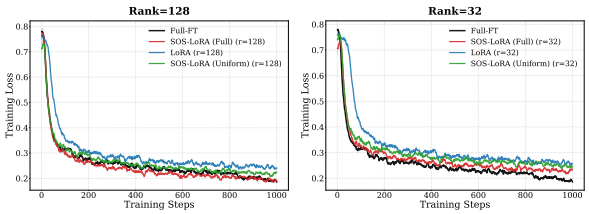


Figure 4: Optimization dynamics on GSM8K. Training loss comparing SOS-LoRA with standard LoRA and the uniform-scale ablation at two rank budgets (r_{tot}=32 and r_{tot}=128). SOS-LoRA exhibits faster early loss reduction and lower final loss.

wise ℓ_2 normalization. As shown in Fig. 2 (left vs. right), full SOS-LoRA exhibits lower off-diagonal similarity, whereas removing $\mathcal{L}_{\text{orth}}$ yields substantially higher off-diagonal values. This pattern is consistent with $\mathcal{L}_{\text{orth}}$ discouraging different experts from converging to redundant input-side directions and promoting better cross-expert separation.

4.3 Scale Separation Analysis

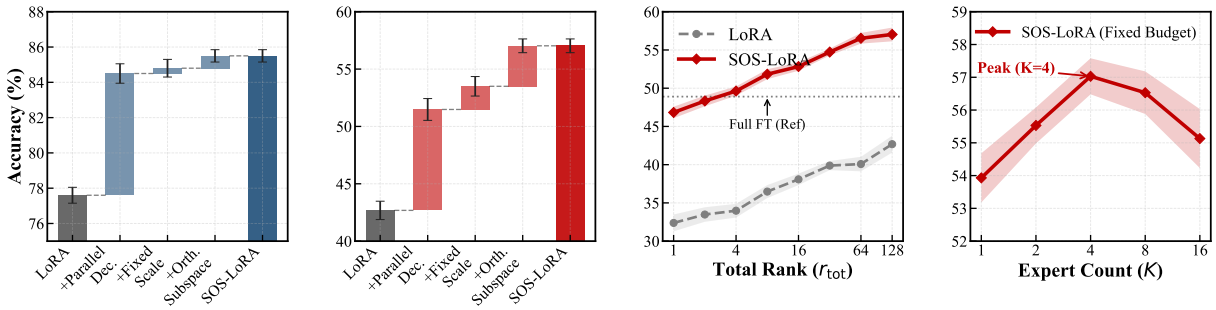
We test whether the pre-specified multi-scale expert scaling promotes scale-separated update magnitudes across experts. Using Llama 2-7B, we compare FULL SOS-LoRA ($\gamma_{\max}$=2.5) with a UNIFORM control ($\gamma_{\max}$=1.0) and measure each expert's relative update magnitude $p_k = \|\Delta W_k\|_F / \sum_{j=1}^{K} \|\Delta W_j\|_F$. As shown in Fig. 3, the left panel indicates that FULL yields a more clearly separated contribution profile across experts, whereas UNIFORM is closer to a balanced allocation around $1/K$. The right panel further shows that the top-1 share $\max_k p_k$ is consistently larger under FULL. These results are consistent with multi-scale scaling separating update magnitudes across experts under a fixed rank budget, which may in turn encourage more differentiated expert roles during training.

4.4 Optimization Dynamics

We examine training dynamics in Fig. 4 by fine-tuning Llama 2-7B on the first 100k MetaMathQA samples (Yu et al., 2023). At both low (r_{tot}=32) and high (r_{tot}=128) rank budgets, SOS-LoRA shows faster early loss reduction and lower final loss than standard LoRA under the same training setup. The UNIFORM ablation reduces this advantage, suggesting that the pre-specified multi-scale scaling scheme contributes materially to the observed acceleration.

4.5 Ablation and Sensitivity Analysis

We analyze SOS-LoRA on Llama 2-7B to assess the contribution of each design component and sensitivity to key design choices and hyperparameters



(a) Ablation (Commonsense) (b) Ablation (GSM8K) (c) Rank Scalability (d) Sensitivity to K

Figure 5: Analysis of SOS-LoRA on Llama 2-7B. (a) Ablation on reasoning and knowledge-intensive benchmarks (standard budget r_{tot}=8): stepwise variants illustrate the contribution of each component. (b) Ablation on GSM8K (high-rank budget r_{tot}=128): under our training setup, performance increases with rank and reaches 57.03%. (c) Rank scalability on GSM8K: SOS-LoRA consistently outperforms LoRA and matches or surpasses Full FT at higher ranks in our setup. (d) Sensitivity to expert count K on GSM8K (fixing r_{tot}=128): performance peaks at K=4, balancing cross-expert diversification and per-expert capacity.

Table 3: Cost–accuracy trade-off on GSM8K and MATH when fine-tuning Llama 2-7B on the first 100k MetaMathQA samples using 2× NVIDIA A40 (48GB). Peak VRAM is the maximum *reserved* memory.

Method	Peak VRAM (GB)↓	Time (wall)↓	GSM8K (%)↑	MATH (%)↑
Full FT†	N/A (OOM)	N/A	48.90	7.48
LoRA	**66.32**	**4:31**	42.68	5.92
SOS-LoRA	67.45	4:52	**57.03**	**10.06**

(Fig. 5). Under fixed rank budgets, the ablations in Fig. 5(a–b) show that parallel expert decomposition provides the largest gain, while multi-scale scaling and orthogonal regularization yield consistent additional improvements, indicating complementary benefits beyond decomposition alone. Fig. 5(c) further shows that SOS-LoRA improves over standard LoRA across a wide range of ranks ($r_{\text{tot}} \in [1, 128]$) and matches or surpasses the Full FT baseline at higher ranks in our setup. Finally, Fig. 5(d) suggests that performance peaks around K=4 at fixed r_{tot}=128: larger K reduces the per-expert rank and may limit per-expert capacity, whereas K=1 removes cross-expert diversification, consistent with a diversification–capacity trade-off.

4.6 Computational Cost Analysis

We benchmark the cost–accuracy trade-off on Llama 2-7B using 2× NVIDIA A40 (48GB) GPUs with DeepSpeed ZeRO-2. We fine-tune on the first 100k MetaMathQA samples (Yu et al., 2023) and evaluate on GSM8K and MATH. Table 3 shows that SOS-LoRA achieves higher accuracy than standard LoRA on both benchmarks (57.03% vs. 42.68% on GSM8K; 10.06% vs. 5.92% on MATH). Relative to LoRA, SOS-LoRA improves by 14.35 points on GSM8K and 4.14 points on MATH. This gain comes with a modest increase in training cost (4:52 vs. 4:31 wall-clock time; 67.45 vs. 66.32 GB peak reserved VRAM), likely attributable primarily to the orthogonal regularizer. Full fine-tuning is not feasible under the same configuration due to OOM; its reported accuracies are obtained with a modified recipe, as indicated by Table 3†. Overall, SOS-LoRA offers a favorable accuracy–cost trade-off under this protocol.

5 Conclusion

We identify shared input-side coupling as a limitation of low-rank adaptation under a fixed rank budget, where a single input-side pathway can couple heterogeneous adaptation signals and induce optimization interference. We propose **Static Orthogonal Subspace LoRA (SOS-LoRA)**, which decomposes each low-rank update into K *static* (always-on, non-routed) experts with a fixed multi-scale scaling scheme and explicit input-side direction diversification via orthogonal initialization and a lightweight regularizer. SOS-LoRA is mergeable into the base model, adding no inference-time parameters or latency after merging. Experiments on both encoder-only (RoBERTa) and decoder-only

(Llama 2/3) backbones show consistent gains over PEFT baselines under matched training budgets, suggesting that explicit input-side direction diversification can provide a practical route to stronger parameter-efficient adaptation.

Limitations

Our evaluation covers both encoder-only (RoBERTa-base) and decoder-only (Llama 2/3) backbones, but it is still limited to a relatively narrow range of model sizes and benchmark suites. Performance in larger models and broader settings (e.g., long-context, multilingual, and tool-use scenarios) remains untested. SOS-LoRA also introduces additional hyperparameters (e.g., expert count K, multi-scale range $\gamma_{\max}$, and regularization weight λ). Although we use a consistent default configuration and provide ablations and sensitivity analyses for key factors, the best settings may still vary with architecture, task, and training recipe. Finally, our analysis relies on geometric proxies (e.g., cross-expert squared-cosine similarity between input-side directions and relative update magnitudes) that capture redundancy and scale separation, but do not directly measure functional specialization. Complementary behavior-level analyses, such as targeted interventions or task-conditioned attribution/probing, would help strengthen these interpretations.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (No.2023YFF0905400), the National Natural Science Foundation of China (No.U2341229) and the Reform Commission Foundation of Jilin Province (No.2024C003).

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*.

AI@Meta. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

AI@Meta. 2024. Llama 3 model card.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Yupeng Chang, Yi Chang, and Yuan Wu. 2026. BA-loRA: Bias-alleviating low-rank adaptation to mitigate catastrophic inheritance in large language models. In *The Fourteenth International Conference on Learning Representations*.

Yupeng Chang, Chenlu Guo, Yi Chang, and Yuan Wu. 2025. Lora-mgpo: Mitigating double descent in low-rank adaptation via momentum-guided perturbation optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 648–659.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, and 1 others. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *Preprint*, arXiv:2104.08691.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and 1 others. 2024. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. *arXiv preprint arXiv:2404.15159*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*.

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

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *SIGKDD*, pages 3505–3506.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten de Rijke, Zhumin Chen, and Jiahuan Pei. 2024. Melora: Mini-ensemble low-rank adapters for parameter-efficient fine-tuning. *arXiv preprint arXiv:2402.17263*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2022. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Shaowen Wang, Linxi Yu, and Jian Li. 2024a. Lora-ga: Low-rank adaptation with gradient approximation. *arXiv preprint arXiv:2407.05000*.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. 2023. Orthogonal subspace learning for language model continual learning. *arXiv preprint arXiv:2310.14152*.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. 2024b. Lora-pro: Are low-rank adapters properly optimized? *Preprint*, arXiv:2407.18242.

Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of lora experts. *arXiv preprint arXiv:2404.13628*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*.

Jiancheng Zhao, Xingda Yu, and Zhen Yang. 2025. Msplora: A multi-scale pyramid low-rank adaptation for efficient model fine-tuning. *arXiv preprint arXiv:2503.21838*.

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

A Appendix

A.1 Method Details of SOS-LoRA

SOS-LoRA replaces a single rank-r_{tot} LoRA update with K *static* (always-on, non-routed) low-rank experts of rank $r' = r_{\text{tot}}/K$. For a linear layer with frozen weight $W_0 \in \mathbb{R}^{m\times n}$ and input activations $X \in \mathbb{R}^{N\times m}$, the SOS-LoRA forward pass is

$$Y = XW_0 + \sum_{k=1}^{K} s_k\, XA_kB_k, \tag{6}$$

where $A_k \in \mathbb{R}^{m\times r'}$, $B_k \in \mathbb{R}^{r'\times n}$, and $s_k > 0$ are fixed expert scales set by the multi-scale schedule in Eq. 3. The parameterization is mergeable: after training, the effective update is $\Delta W = \sum_{k=1}^{K} s_k A_k B_k$, which can be added to W_0 to obtain a standard dense weight for inference, incurring no additional inference-time parameters or latency after merging. We use cross-expert orthogonal initialization with $B_k = 0$, so that $\Delta W = 0$ at initialization while the A_k start from diverse input-side directions. The cross-expert diversity regularizer is the squared-cosine penalty in Eq. 5.

A.2 Algorithmic Summary of SOS-LoRA

For clarity and reproducibility, we provide a high-level outline of SOS-LoRA for a single linear layer. The same procedure is applied to every adapted linear projection while keeping the pretrained weights frozen; all steps correspond to the definitions in Section 3. The procedure is summarized in Algorithm 1.

Algorithm 1 SOS-LoRA training and merging for a single linear layer

Require: Frozen $W_0 \in \mathbb{R}^{m\times n}$; total rank r_{tot}; number of experts K ($r' = r_{\text{tot}}/K$); fixed scales $\{s_k\}$ (Eq. 3)

Initialize: when $r_{\text{tot}} = Kr' \le m$, sample $G \in \mathbb{R}^{m\times Kr'}$, compute thin QR $G = QR$, and split $Q = [Q_1, \ldots, Q_K]$

  Set $A_k \leftarrow Q_k$ and $B_k \leftarrow 0$ for $k = 1, \ldots, K$ ▷ $\Delta W = 0$ at initialization

for each training step **do**

  Forward: $Y \leftarrow XW_0 + \sum_{k=1}^{K} s_k\, XA_kB_k$

  Objective: $\mathcal{L} \leftarrow \mathcal{L}_{\text{task}}(Y) + \mathcal{L}_{\text{orth}}$ ▷ $\mathcal{L}_{\text{orth}}$ is defined in Eq. 5

  Update: update $\{A_k, B_k\}_{k=1}^{K}$ with the chosen optimizer; keep W_0 frozen

end for

Merge for inference: $\Delta W \leftarrow \sum_{k=1}^{K} s_k A_k B_k$; $W \leftarrow W_0 + \Delta W$; discard adapter parameters

A.3 Training Setup and Hyperparameters

All experiments are implemented in PyTorch with Hugging Face Transformers and the PEFT library. Unless otherwise noted, SOS-LoRA is applied to all linear projections. For decoder-only LLMs (the Llama family), we adapt the attention projections (`q,k,v,o`) and MLP projections (`up,gate,down`); for RoBERTa-base, we adapt the self-attention and feed-forward linear layers. Unless otherwise specified, we use r_{tot}=8 per adapted matrix with K=4 experts (r'=2), LoRA dropout rate 0.05, $\gamma_{\max}$=2.5 in Eq. 3, and λ=0.01 in Eq. 5. We optimize with AdamW with a linear learning-rate schedule and warmup ratio 0.1. For RoBERTa-base on GLUE, we train for 3 epochs with learning rate 2×10^{-4}, batch size 32, and max sequence length 512. Unless otherwise stated, Llama experiments use 3 epochs with learning rate 3×10^{-5}, per-device batch size 4 with 8 gradient accumulation steps (effective batch size 32), and max sequence length 2048. Unless stated otherwise, reported results are averaged over three independent runs with different random seeds; we report mean ± standard deviation where applicable. For sweeps over r_{tot} and/or K, we keep all other settings fixed to isolate the effect of varying rank capacity and expert granularity.

A.4 Datasets, Splits, and Metrics

We use public benchmarks with their official splits. Reasoning and knowledge-intensive evaluation includes BoolQ, PIQA, SocialIQA, HellaSwag, WinoGrande, ARC-E/ARC-C, and OpenBookQA, evaluated under a unified prompting and scoring pipeline with accuracy as the metric. NLU evaluation uses GLUE with task-specific metrics following the benchmark conventions (Wang et al., 2018): CoLA uses Matthews correlation, STS-B uses correlation, SST-2/QNLI/RTE use accuracy, and MRPC/QQP are conventionally reported with both F1 and accuracy. In Table 2, we report accuracy for MRPC and QQP (while retaining F1 in our released logs/configs), report MNLI matched accuracy (MNLI-m), and report the simple average over the eight tasks shown (excluding WNLI). Math reasoning evaluation uses GSM8K and MATH: GSM8K is evaluated by exact-match

accuracy on the final answer, and MATH is evaluated by final-answer accuracy under the same post-processing rules across methods (Cobbe et al., 2021; Hendrycks et al., 2021).

A.5 Baselines and Configuration Matching

We compare SOS-LoRA against Full FT as a reference and LoRA as the primary PEFT baseline. We additionally include representative LoRA variants reported in the main results tables: DoRA, AdaLoRA, DyLoRA, MELoRA, PiSSA, DeltaLoRA, MixLoRA, and MSP-LoRA. For fair comparison, we match (i) backbone checkpoints, (ii) training data and the number of steps/epochs, (iii) the set of adapted modules, unless a baseline by design constrains insertion locations, and (iv) the total low-rank budget per adapted matrix whenever a direct rank match is supported. When a baseline introduces additional components (e.g., routing or extra parameters), we run it under the same training pipeline and overall optimization budget, while following the authors' recommended defaults for baseline-specific hyperparameters. All configurations are recorded in our released files.

A.6 Mechanistic Analyses: Definitions and Computation

Cross-expert input-side direction similarity. For each adapted module, we compute a $K \times K$ matrix with entries $\frac{1}{r'^2}\|\tilde{A}_k^\top \tilde{A}_\ell\|_F^2$, where $\tilde{A}_k$ is obtained by ℓ_2-normalizing each column of A_k. This quantity equals the mean squared cosine similarity over all cross-expert column pairs and matches the quantity used in the regularizer in Eq. 5. We report both module-wise matrices and their average across all adapted modules (224 for Llama 2-7B under our adaptation scope), and visualize them as heatmaps (Fig. 2).

Per-expert relative update magnitude. For each adapted module, we compute $\Delta W_k = s_k A_k B_k$ and define $p_k = \|\Delta W_k\|_F / \sum_{j=1}^{K} \|\Delta W_j\|_F$. We plot the distribution of p_k across modules together with the distribution of $\max_k p_k$ to characterize scale separation (Fig. 3).

Loss dynamics. For the training curves in Fig. 4, we fine-tune Llama 2-7B on the first 100k MetaMathQA samples and log training loss at fixed intervals under identical optimization settings across methods, using the same token budget and sequence length. Curves show the mean across seeds.

A.7 Efficiency Measurement Protocol

For the cost–accuracy study (Table 3), we fine-tune Llama 2-7B on the first 100k MetaMathQA samples using 2× NVIDIA A40 (48GB) GPUs with DeepSpeed ZeRO-2. Peak VRAM is measured as the maximum *reserved* memory (e.g., `torch.cuda.max_memory_reserved`) summed over the two devices during training. Wall-clock time is measured end-to-end under the same training budget and logging frequency across compared methods. Full FT is marked with † because it requires a modified recipe to avoid OOM; we therefore report its accuracies only as a reference, and its cost is not directly comparable under the same configuration.

A.8 ChatGPT Reference Evaluation

We include ChatGPT in Table 1 only as a reference. We evaluate it under the same prompting and scoring format used in our unified evaluation pipeline, with fixed prompt templates and deterministic decoding when supported. The exact prompts, scripts, and post-processing rules are included in our release to enable reproduction of the evaluation procedure, subject to model/version changes by the provider. We do not use ChatGPT outputs for training, validation, or hyperparameter selection.

A.9 Reproducibility Statement

We provide the public code repository, training/evaluation scripts, and SOS-LoRA adapter checkpoints under an Apache 2.0 license. Our implementation is based on PyTorch, Hugging Face Transformers, and PEFT; experiments additionally use DeepSpeed for the ZeRO-2 efficiency study. We use publicly available pretrained checkpoints (e.g., Llama 2/3 and RoBERTa-base) and standard benchmarks (e.g., GLUE, GSM8K, MATH) with official splits. Section 4 and Appendix A.3 report key hyperparameters, including adaptation scope, ranks, K, scaling, regularization, learning rates, batch sizes, sequence lengths, epochs, and optimizer schedule. Unless stated otherwise, all reported numbers are mean ± standard deviation over three independent runs with different random seeds.

A.10 LLM Usage Statement

We used a large language model (LLM) only for language polishing and proofreading of the manuscript text. The LLM was not used to design

SOS-LoRA, develop algorithms, generate experimental results, or make methodological or scientific decisions. All technical contributions, experiments, and conclusions are our own, and we take full responsibility for the content of this paper.

A.11 Responsible NLP Research Checklist

We follow the ACL community's Responsible NLP Research guidelines and checklist and include the completed checklist in the final version, as required by the venue. Our experiments are conducted on widely used public benchmarks and standard model checkpoints; we report averaged results over multiple seeds and provide implementation details to support reproducibility. Although our work focuses on parameter-efficient adaptation rather than new data collection, downstream deployment of fine-tuned models should still consider dataset biases, evaluation limitations, and potential misuse, consistent with the checklist guidance.