ProLoSA: Improving Compressed LoRA with Sparse Residual Adaptation

Anonymous ACL submission

Abstract

Low-rank adaptation (LoRA) is a widely used parameter-efficient fine-tuning method, yet its update space remains redundant. Recent compressed LoRA methods reduce trainable parameters by reconstructing LoRA updates from a low-dimensional latent space, but their globally shared compression may over-constrain adaptation and miss locally important directions. This work proposes **Pro**jected **Lo**RA with **S**parse **A**dapter (ProLoSA), which represents the full LoRA parameter vector as the sum of a globally projected latent parameter and a routed sparse residual, preserving the regularization benefit of compression while restoring local flexibility. We further provide a bias–variance analysis showing that compressed LoRA can improve generalization by reducing estimation variance when the downstream update is well aligned with the compressed subspace, while the sparse residual branch reduces the resulting projection bias by recovering off-subspace components. Experiments on synthetic validation and downstream benchmarks show that ProLoSA achieves a better performance–parameter trade-off than compressed LoRA baselines under comparable trainable-parameter budgets.

1 Introduction

Large language models (LLMs) have achieved strong performance across a wide range of natural language understanding and generation tasks, but adapting them through full fine-tuning remains computationally and storage intensive. Parameter-efficient fine-tuning (PEFT) addresses this issue by updating only a small number of task-specific parameters while keeping the pretrained backbone frozen. Among PEFT methods, Low-Rank Adaptation (LoRA) has become a widely used approach due to its simplicity, strong empirical performance, and zero additional inference latency after weight merging (Hu et al., 2022).

Despite its effectiveness, standard LoRA is still not fully efficient in how it parameterizes task-specific updates. Prior work suggests that the effective degrees of freedom required for adaptation can be much smaller than the ambient parameter space (Aghajanyan et al., 2021), while different layers and modules often contribute unequally to downstream performance (Zhang et al., 2023). These observations indicate that LoRA updates contain substantial redundancy, but also that downstream adaptation is non-uniform and may require flexibility in a small number of important directions.

Recent compressed and shared LoRA methods exploit this redundancy by reconstructing the full LoRA parameter vector from fewer latent, shared, or structured trainable parameters. Representative examples include Tied-LoRA (Renduchintala et al., 2024), VeRA (Kopiczko et al., 2024), VB-LoRA (Li et al., 2024), LoRA-XS (Bałazy et al., 2025), and Uni-LoRA (Li et al., 2025). In this paper, we refer to this family of methods as *compressed LoRA*. These methods show that the key issue is not only how many trainable parameters are used, but also how they are organized and mapped back to the original LoRA space.

However, existing compressed LoRA methods mainly rely on global parameter sharing. Their constrained reconstruction spaces can reduce redundant degrees of freedom, but may also miss layer-specific or coordinate-specific directions when the downstream update is poorly aligned with the compressed subspace. Sparse and hybrid adaptation methods provide a complementary source of local flexibility, yet they are typically not designed as corrective mechanisms within globally compressed LoRA reconstruction. This leads to a central question: *how can we preserve the regularization and efficiency benefits of global compression while restoring local flexibility for important directions that compression may miss?*

We answer this question by proposing **Pro**jected **Lo**RA with **S**parse **A**dapter (**ProLoSA**). ProLoSA

represents the full LoRA parameter vector as the sum of a projected compressed branch and a routed sparse residual branch. The projected branch follows the Uni-LoRA-style projection framework and reconstructs LoRA parameters from a low-dimensional latent vector, capturing globally shared adaptation structure. The sparse residual branch assigns a small number of additional trainable coefficients to high-sensitivity coordinates, providing local corrective directions that are difficult to express through purely shared compression. In this way, ProLoSA preserves the variance-reduction benefit of compressed adaptation while reducing the projection bias caused by over-constrained reconstruction.

We further provide a bias–variance analysis to explain why compressed LoRA can outperform classical LoRA in certain regimes rather than merely approximate it. The analysis shows that compression can improve generalization by lowering estimation variance when the downstream update is well aligned with the compressed subspace, but it may introduce projection bias when important off-subspace directions are present. The sparse residual branch directly addresses this limitation by enlarging the feasible adaptation space with a small number of coordinate-wise corrective parameters.

Our contributions are summarized as follows:

- We propose ProLoSA, a compressed LoRA framework that combines a projected latent branch with a routed sparse residual branch to balance global parameter sharing and local corrective flexibility.

- We provide a bias–variance explanation of compressed LoRA, showing that low-dimensional reconstruction is advantageous when the reduction in estimation variance outweighs the projection bias, and that ProLoSA improves over compressed-only LoRA when sparse residual correction removes more projection bias than the additional variance it introduces.

- We show through synthetic validation and downstream experiments that ProLoSA achieves a favorable performance–parameter trade-off compared with standard LoRA and compressed LoRA baselines.

2 Related Work

LoRA and compressed parameterizations. Parameter-efficient fine-tuning (PEFT) adapts pretrained language models by updating a small number of task-specific parameters while keeping the backbone frozen. Among PEFT methods, LoRA (Hu et al., 2022) represents weight updates with low-rank matrices and has become a widely used approach due to its effectiveness and mergeability. Subsequent studies improve LoRA from different perspectives: AdaLoRA (Zhang et al., 2023) adaptively allocates rank according to parameter importance, while DoRA (Liu et al., 2024) improves low-rank adaptation through weight decomposition. A parallel line further compresses the LoRA parameter space by sharing or reconstructing trainable parameters. Tied-LoRA (Renduchintala et al., 2024) uses cross-layer tying, VeRA (Kopiczko et al., 2024) learns lightweight scaling vectors over frozen random matrices, VB-LoRA (Li et al., 2024) introduces a global vector bank, and LoRA-XS (Bałazy et al., 2025) inserts a small trainable matrix between frozen low-rank factors. Uni-LoRA (Li et al., 2025) formulates such methods under a global projection view, where full LoRA parameters are reconstructed from a shared low-dimensional vector. We refer to this family as compressed LoRA methods and adopt the normalized one-hot Uni-LoRA projection as the compressed branch in our implementation. While these methods exploit global redundancy, they do not explicitly model sparse localized residual directions that may be important for downstream adaptation.

Sparse and hybrid adaptation. Sparsity provides a complementary inductive bias for efficient adaptation. SparseAdapter (He et al., 2022) improves adapter tuning by pruning adapter parameters, while sparse fine-tuning methods such as SIFT (Song et al., 2024) and SpIEL (Ansell et al., 2024) directly optimize a sparse subset of parameter updates. Hybrid approaches combine sparse and low-rank structures; for example, RoSA (Nikdan et al., 2024) jointly trains low-rank and highly sparse components on top of frozen pretrained weights. ProLoSA is related to this hybrid direction, but differs in its starting point and role of sparsity: rather than adding sparse corrections to a standard low-rank update, we introduce a sparse residual branch as a corrective mechanism for globally compressed LoRA reconstruction. This allows ProLoSA to retain the parameter-sharing benefit of

compressed LoRA while restoring local flexibility through sparse residual adaptation.

3 Proposed Methodology

We present ProLoSA by first revisiting compressed LoRA as a restricted estimation problem in the full LoRA parameter space. This view makes explicit the central trade-off: a projected low-dimensional parameterization can reduce estimation variance, but may introduce approximation error when important downstream directions lie outside the projected subspace. ProLoSA is designed to address this trade-off by retaining the globally shared projected branch while adding a small routed sparse residual branch for local correction. We then detail the resulting parameterization, warmup-based support selection, and training algorithm.

3.1 A Bias–Variance View of Compressed LoRA

We first view LoRA-style adaptation in the full LoRA parameter space. Let $\theta_D \in \mathbb{R}^D$ denote the vectorized collection of all trainable LoRA factor entries across the adapted layers. Compressed LoRA methods do not optimize θ_D directly. Instead, they learn a lower-dimensional latent parameter $\theta_d \in \mathbb{R}^d$ with $d \ll D$ and reconstruct the full LoRA parameter vector from it. For analysis, we focus on the linear projected form

$$\theta_D = P\theta_d, \tag{1}$$

where $P \in \mathbb{R}^{D\times d}$ is a fixed reconstruction matrix. Classical LoRA corresponds to the special case $d = D$ and $P = I$. This projected form covers the Uni-LoRA-style parameterization used in our compressed branch and provides a convenient abstraction for analyzing the effect of compression.

The key question is why restricting the adaptation space can improve generalization rather than merely reduce expressiveness. Let

$$\begin{aligned} R(\theta) &= \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[\ell(f(x;\theta), y)\big], \\ \widehat{R}_n(\theta) &= \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i;\theta), y_i) \end{aligned} \tag{2}$$

denote the population and empirical risks (Vapnik, 1998; Hastie et al., 2009; van der Vaart, 1998) on the downstream data distribution, and let

$$\theta^\star = \arg\min_{\theta\in\mathbb{R}^D} R(\theta), \qquad \hat{\theta}_L = \arg\min_{\theta\in\mathbb{R}^D} \widehat{R}_n(\theta) \tag{3}$$

denote the full-space population optimum and the empirical minimizer of classical LoRA, respectively. Around $\theta^\star$, we use the standard local quadratic approximation

$$R(\theta) \approx R(\theta^\star) + \frac{1}{2}(\theta-\theta^\star)^\top H(\theta-\theta^\star), \qquad H \succeq 0, \tag{4}$$

where $H = \nabla^2 R(\theta^\star)$ denotes the local curvature. This analysis is intended as a local approximation around the full-space population optimum, rather than a claim of global convexity for neural-network fine-tuning.

In the locally centered case, where $\hat{\theta}_L$ concentrates around $\theta^\star$ and its finite-sample centering bias is negligible, the expected excess risk is

$$\mathbb{E}[R(\hat{\theta}_L)] - R(\theta^\star) \approx \frac{1}{2}\mathrm{tr}(H\Sigma_L), \tag{5}$$

where Σ_L is the covariance of the estimator. Thus, classical LoRA has no approximation bias with respect to the full LoRA parameter space, but may incur high estimation variance because it optimizes in the full D-dimensional space. A more general decomposition including finite-sample and optimization bias is provided in Appendix A.1.

For a compressed parameterization, optimization is restricted to the projected subspace $\mathcal{S}_U = \mathrm{col}(P)$. The corresponding restricted population and empirical minimizers are

$$\theta_U^\star = \arg\min_{\theta\in\mathcal{S}_U} R(\theta), \qquad \hat{\theta}_U = \arg\min_{\theta\in\mathcal{S}_U} \widehat{R}_n(\theta). \tag{6}$$

Writing $\hat{\theta}_U = P\hat{z}_U$ and $\theta_U^\star = Pz_U^\star$, define

$$\Sigma_z = \mathrm{Cov}(\hat{z}_U - z_U^\star). \tag{7}$$

If the empirical compressed estimator is written locally around $\theta_U^\star$ with latent covariance Σ_z, its expected excess risk becomes

$$\begin{aligned} \mathbb{E}[R(\hat{\theta}_U)] - R(\theta^\star) \approx\ & \underbrace{\frac{1}{2}(\theta_U^\star - \theta^\star)^\top H(\theta_U^\star - \theta^\star)}_{\text{projection bias}} \\ & + \underbrace{\frac{1}{2}\mathrm{tr}(HP\Sigma_z P^\top)}_{\text{estimation variance}}. \end{aligned} \tag{8}$$

Therefore, compressed LoRA can outperform classical LoRA when the variance reduction obtained by restricting the optimization space outweighs the additional projection bias:

$$(\theta_U^\star-\theta^\star)^\top H(\theta_U^\star-\theta^\star) + \mathrm{tr}(HP\Sigma_z P^\top) < \mathrm{tr}(H\Sigma_L). \tag{9}$$

This explains why stronger compression does not necessarily imply weaker adaptation: the projected parameterization can act as an inductive bias by removing redundant or noisy directions.

However, aggressive compression may introduce projection bias when important downstream adaptation directions lie outside $\mathrm{col}(P)$. ProLoSA addresses this limitation by augmenting the compressed branch with a sparse residual branch. Let $I = \{i_1, \ldots, i_K\} \subseteq \{1, \ldots, D\}$ denote a sparse support of size K, and let

$$E_I = [e_{i_1}, \ldots, e_{i_K}] \in \mathbb{R}^{D \times K} \tag{10}$$

be the corresponding coordinate-selection matrix. For a fixed support I, the augmented ProLoSA space is

$$\mathcal{S}_{U,I} = \{Pz + E_I\alpha \mid z \in \mathbb{R}^d, \alpha \in \mathbb{R}^K\} = \mathrm{col}([P, E_I]). \tag{11}$$

Since $\mathcal{S}_U = \mathrm{col}(P) \subseteq \mathcal{S}_{U,I}$, the sparse residual branch enlarges the feasible adaptation space.

Let

$$\theta^\star_{U,I} = \arg\min_{\theta \in \mathcal{S}_{U,I}} R(\theta), \qquad \hat{\theta}_{U,I} = \arg\min_{\theta \in \mathcal{S}_{U,I}} \widehat{R}_n(\theta) \tag{12}$$

denote the population and empirical minimizers in the ProLoSA feasible space. We define the compressed-only and ProLoSA approximation biases as

$$\begin{aligned} \mathsf{Bias}_U &:= \frac{1}{2}(\theta^\star_U - \theta^\star)^\top H(\theta^\star_U - \theta^\star), \\ \mathsf{Bias}_{U,I} &:= \frac{1}{2}(\theta^\star_{U,I} - \theta^\star)^\top H(\theta^\star_{U,I} - \theta^\star). \end{aligned} \tag{13}$$

Proposition 1. For any fixed sparse support I, the best attainable local approximation bias of ProLoSA is no larger than that of the compressed-only parameterization:

$$\mathsf{Bias}_{U,I} \leq \mathsf{Bias}_U. \tag{14}$$

The result follows from the inclusion $\mathcal{S}_U \subseteq \mathcal{S}_{U,I}$: ProLoSA minimizes the same local quadratic approximation error over a larger feasible set than compressed-only LoRA. A formal proof is provided in Appendix A.2. This monotonicity result concerns only the best attainable approximation bias. The overall expected excess risk may still increase if the sparse branch introduces more estimation variance than the projection bias it removes.

This proposition captures the complementary role of the two branches. The projected branch reduces estimation variance through global low-dimensional sharing, while the sparse residual branch allocates a small number of additional parameters to directions that may be poorly represented by the projected subspace. The improvement is beneficial when the recovered projection bias is larger than the additional variance introduced by the sparse coefficients. Detailed derivations, including the relaxed bias–variance decomposition, the isotropic case, and the corresponding ProLoSA bias–variance condition, are provided in Appendices A.1–A.4.

3.2 ProLoSA Parameterization

We now describe the parameterization of ProLoSA. Let $\mathcal{L}$ denote the set of adapted linear layers in a pretrained model with frozen weights W_0. For each layer $\ell \in \mathcal{L}$, LoRA introduces a low-rank update

$$\Delta W_\ell = B_\ell A_\ell, \tag{15}$$

where $A_\ell \in \mathbb{R}^{r_\ell \times d_\ell^{\mathrm{in}}}$ and $B_\ell \in \mathbb{R}^{d_\ell^{\mathrm{out}} \times r_\ell}$. We vectorize all LoRA factors into a full LoRA parameter vector

$$\theta_D = \mathrm{vec}\big(\{A_\ell, B_\ell\}_{\ell \in \mathcal{L}}\big) \in \mathbb{R}^D. \tag{16}$$

Instead of directly optimizing θ_D, ProLoSA reconstructs it as

$$\theta_D = P\theta_d + R\alpha, \tag{17}$$

where $\theta_d \in \mathbb{R}^d$ is the compressed latent parameter, $P \in \mathbb{R}^{D \times d}$ is a fixed projection matrix, $R \in \{0,1\}^{D \times K}$ routes sparse residual coefficients to selected full-space coordinates, and $\alpha \in \mathbb{R}^K$ contains the trainable sparse coefficients. The total number of trainable parameters is therefore

$$B = d + K, \tag{18}$$

with $d \ll D$ and $K \ll D$.

We instantiate P using the normalized one-hot projection of Uni-LoRA (Li et al., 2025). Let $\phi : \{1, \ldots, D\} \to \{1, \ldots, d\}$ assign each full-space LoRA coordinate to a latent coordinate, and let $c_j = |\{i : \phi(i) = j\}|$ be the number of coordinates assigned to latent coordinate j. The projection matrix is defined as

$$P_{ij} = \begin{cases} c_j^{-1/2}, & \text{if } j = \phi(i), \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

Thus, each row of P has exactly one nonzero entry, and the columns are normalized so that $P^\top P = I_d$

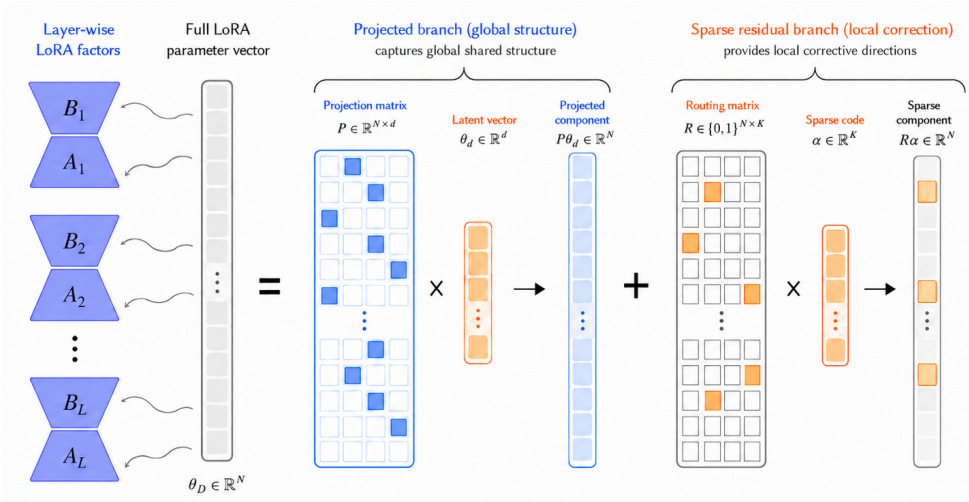

Figure 1: Overview of **ProLoSA**. The full LoRA parameter vector is reconstructed as $\theta_D = P\theta_d + R\alpha$, where the projected branch captures globally shared adaptation structure and the sparse residual branch restores local corrective directions on selected coordinates.

when all latent coordinates are nonempty. The assignment is sampled once at initialization and then kept fixed during training.

This construction partitions the D LoRA coordinates into d shared groups. For coordinate i, the projected reconstruction is

$$(P\theta_d)_i = c_{\phi(i)}^{-1/2}\theta_{d,\phi(i)}. \tag{20}$$

Hence, the projected branch ties multiple LoRA coordinates through shared latent parameters, while the sparse residual branch $R\alpha$ provides direct coordinate-wise correction for selected high-sensitivity directions. After reconstructing θ_D, we unpack its corresponding slices back into layer-wise LoRA factors:

$$(A_\ell, B_\ell) = \mathcal{U}_\ell(\theta_D), \qquad \Delta W_\ell = B_\ell A_\ell. \tag{21}$$

Since the final update is still represented as standard LoRA factors, it can be merged into the frozen backbone after training and introduces no additional inference latency.

3.3 Routed Sparse Adapter with Warmup Support Selection

The sparse residual branch requires selecting a small support in the full LoRA parameter space. Rather than learning a dense residual with D additional parameters, ProLoSA activates only K coordinates selected after a compressed warmup stage.

Let K be the sparse budget. The routing matrix $R \in \{0,1\}^{D\times K}$ contains one nonzero entry per column, equivalently assigning each sparse coefficient to a full-space coordinate through a mapping $\pi : \{1,\dots,K\} \to \{1,\dots,D\}$. The sparse residual is

$$R\alpha = \sum_{j=1}^{K} \alpha_j e_{\pi(j)}, \tag{22}$$

and therefore has at most K nonzero coordinates.

During warmup, the sparse branch is inactive and the model uses only the compressed reconstruction

$$\theta_D^{(t)} = P\theta_d^{(t)}. \tag{23}$$

At each warmup step t, we compute the mini-batch loss $\mathcal{L}^{(t)}$ and the gradient $g^{(t)} = \nabla_{\theta_D^{(t)}}\mathcal{L}^{(t)}$. We accumulate a SNIP-style sensitivity score for each full-space coordinate:

$$s_i = \sum_{t=1}^{T_{\mathrm{w}}} \left|\theta_{D,i}^{(t)} g_i^{(t)}\right|, \tag{24}$$

where T_{w} is the number of warmup steps. The score measures the first-order effect of perturbing coordinate i, following SNIP-style pruning criteria (Lee et al., 2019; He et al., 2022). Appendix A.5 provides the corresponding first-order derivation and explains its connection to sparse support selection.

After warmup, ProLoSA selects the top-K coordinates

$$\mathcal{I} = \mathrm{TopK}\left(\{s_i\}_{i=1}^{D}, K\right), \tag{25}$$

and constructs

$$R = [e_{i_1}, e_{i_2}, \dots, e_{i_K}], \qquad \mathcal{I} = \{i_1, \dots, i_K\}. \tag{26}$$

The support is fixed for the remaining training steps, and the sparse coefficients are initialized as $\alpha = 0$ so that activating the residual branch does not abruptly change θ_D. Then θ_d and α are jointly optimized under the reconstruction $\theta_D = P\theta_d + R\alpha$.

This fixed-support design avoids the instability of repeatedly changing the parameterization while still allocating additional degrees of freedom to coordinates that the compressed warmup identifies as highly sensitive.

3.4 Training Objective

Given the parameterization in Eq. (17), ProLoSA is trained in two stages. First, a compressed warmup stage optimizes only θ_d and accumulates the sensitivity scores in Eq. (24) to select the sparse support. Second, with the routing matrix R fixed and α initialized to zero, ProLoSA jointly optimizes θ_d and α under

$$\theta_D = P\theta_d + R\alpha. \tag{27}$$

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ be the downstream training set. After support selection, the training objective is

$$\min_{\theta_d,\alpha} \frac{1}{N} \sum_{i=1}^{N} \ell\big(f(x_i; W_0, \theta_d, \alpha), y_i\big), \tag{28}$$

where the pretrained backbone W_0 is frozen. At each step, θ_D is reconstructed and unpacked into layer-wise LoRA factors, while P and R remain fixed. After training, the resulting LoRA updates are merged into the backbone weights, so ProLoSA has the same inference-time form as standard merged LoRA. The full training procedure is summarized in Appendix A.6.

4 Experiments

We evaluate ProLoSA on synthetic validation, GLUE natural language understanding, mathematical reasoning, and instruction tuning. The synthetic task tests the bias–variance prediction in Section 3.1, while the downstream benchmarks assess whether the proposed hybrid parameterization improves practical PEFT under tight parameter budgets. All experiments were conducted on NVIDIA L20 GPUs.

4.1 Experimental Setup

For GLUE, we use RoBERTa-base and RoBERTa-large and report standard task metrics. For mathematical reasoning, we evaluate decoder-only LLMs on GSM8K and MATH using exact-match accuracy. For instruction tuning, we train on Cleaned Alpaca and evaluate with MT-Bench. We compare against LoRA and representative compressed PEFT baselines under matched or reported trainable-parameter budgets. Full details are provided in Appendix B.

4.2 Controlled Bias–Variance Validation

We first validate the theoretical prediction using a synthetic teacher–student linear regression task. The ground-truth update is constructed as

$$\theta^\star = Pz^\star + \gamma s_\perp, \tag{29}$$

where P is the normalized one-hot projection used by the compressed branch, $s_\perp \perp \mathrm{col}(P)$, and γ controls the mismatch between the true update and the compressed subspace. We compare an unconstrained LoRA-space estimator, a Uni-LoRA-style projected estimator, and ProLoSA. In ProLoSA, the sparse support is selected using the same warmup SNIP score as in the proposed method, rather than an oracle support.

Figure 2 confirms the predicted bias–variance trade-off. In the small-sample regime, LoRA suffers from large estimation variance because it optimizes in the full LoRA parameter space. Uni-LoRA substantially lowers the risk by restricting estimation to the compressed subspace, but its risk increases as γ grows due to projection bias. ProLoSA further augments the compressed subspace with SNIP-selected sparse residual coordinates. This allows ProLoSA to recover part of the off-subspace signal while preserving most of the variance-reduction benefit of compression. The improvement is most evident when the sparse residual branch removes more projection bias than the additional variance introduced by the sparse coefficients. For example, at $n = 256$ and $\gamma = 2$, ProLoSA reduces the population excess risk from 4.524 to 2.661 compared with Uni-LoRA.

4.3 Main Results

4.3.1 Results on GLUE

Table 1 reports the results on GLUE (Wang et al., 2018). Overall, **ProLoSA** achieves a stronger performance–parameter trade-off than standard LoRA (Hu et al., 2022) and compressed PEFT baselines under matched or smaller trainable budgets. Compared with Uni-LoRA (Li et al., 2025), **ProLoSA** improves the average score from 85.3 to

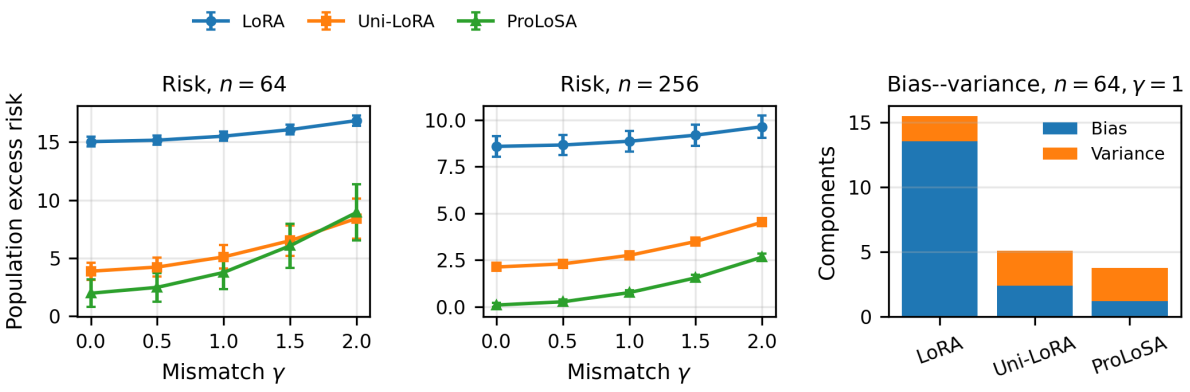

Figure 2: Synthetic validation of the bias–variance theory. Left and middle: population excess risk as the subspace mismatch γ increases under different sample sizes. Right: empirical bias–variance decomposition at $n = 64$ and $\gamma = 1$. ProLoSA uses warmup SNIP scores to select sparse residual coordinates, mitigating the projection bias of compressed adaptation when the recovered off-subspace signal outweighs the additional sparse-branch variance.

85.5 on RoBERTa-base and from 88.3 to 88.7 on RoBERTa-large while using the same number of trainable parameters. These results suggest that the sparse residual branch can recover task-specific directions that are difficult to represent using the compressed projection alone, while preserving the parameter efficiency of compressed LoRA.

4.3.2 Results on mathematical reasoning

Using Gemma-7B (Gemma Team, 2024) as the backbone, Table 2 reports results on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). ProLoSA shows its main advantage on the harder MATH benchmark, improving over Uni-LoRA from 28.94 to 29.90 under the same 0.52M trainable-parameter budget and outperforming other compressed baselines. Although it is slightly lower than Uni-LoRA on GSM8K and full LoRA still achieves the best MATH score with 200M parameters, these results suggest that sparse residual correction helps recover adaptation directions that pure compression may miss in more demanding reasoning tasks.

4.3.3 Results on instruction tuning

Table 3 summarizes the results on instruction tuning. We focus on this setting because it is one of the most practically relevant scenarios for PEFT, while also being sensitive to both optimization stability and adaptation capacity. Under the same QLoRA-based training framework (Dettmers et al., 2023), ProLoSA achieves superior MT-Bench (Zheng et al., 2023) performance while using substantially fewer trainable parameters than standard LoRA (Hu et al., 2022) and strong PEFT baselines. This suggests that the sparse residual branch recovers critical task-specific degrees of freedom that are lost in overly rigid compressed parameterizations, without sacrificing the regularization benefit of global sharing.

4.4 Ablation Study

We conduct ablation studies on CoLA and MRPC to examine three design choices in ProLoSA: the budget allocation between the projected and sparse branches, the sparse support selection strategy, and the warmup length. All ablations are conducted under matched trainable-parameter budgets. Table 4 summarizes the main results, and the full budget sweep is provided in Appendix B.5.

The budget allocation results show that neither pure compression nor pure sparsity is optimal. Hybrid allocations outperform the sparse-only variant by a large margin and also improve over compressed-only in at least one task, indicating that the projected branch and sparse residual branch provide complementary capacity. The support-selection results further show that warmup SNIP-style routing outperforms both random support and accumulated-gradient support under the same d and K, suggesting that the sparse residual branch is most effective when assigned to high-sensitivity coordinates rather than arbitrary coordinates. Finally, a short warmup is sufficient: 10% warmup gives the best performance, while a longer 20% warmup degrades both tasks, likely because it reduces the effective joint-optimization phase after the sparse branch is activated.

Table 1: Results with RoBERTa$_{\text{base}}$ and RoBERTa$_{\text{large}}$ on GLUE (Wang et al., 2018). We report Matthews correlation for CoLA, Pearson correlation for STS-B, and accuracy for all other tasks. The median performance over 5 runs with different random seeds is reported. Baseline results are taken from Uni-LoRA (Li et al., 2025).

Backbone	Method	Trainable Params	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
BASE	FT	125M	94.8	90.2	63.6	92.8	78.7	91.2	85.2
	LoRA	0.295M	**95.1**$_{\pm 0.2}$	89.7$_{\pm 0.7}$	63.4$_{\pm 1.2}$	**93.3**$_{\pm 0.3}$	**86.6**$_{\pm 0.7}$	**91.5**$_{\pm 0.2}$	**86.6**
	VeRA	0.043M	94.6$_{\pm 0.1}$	89.5$_{\pm 0.5}$	**65.6**$_{\pm 0.8}$	91.8$_{\pm 0.2}$	78.7$_{\pm 0.7}$	90.7$_{\pm 0.2}$	85.2
	Tied-LoRA	0.043M	94.4$_{\pm 0.5}$	88.5$_{\pm 1.0}$	61.9$_{\pm 1.6}$	92.0$_{\pm 0.4}$	76.2$_{\pm 1.3}$	89.8$_{\pm 0.3}$	83.8
	VB-LoRA	0.075M†	94.4$_{\pm 0.2}$	89.5$_{\pm 0.5}$	63.3$_{\pm 0.7}$	92.2$_{\pm 0.2}$	82.3$_{\pm 1.3}$	90.8$_{\pm 0.1}$	85.4
	FourierFT	0.024M	94.2$_{\pm 0.3}$	**90.0**$_{\pm 0.8}$	63.8$_{\pm 1.6}$	92.2$_{\pm 0.1}$	79.1$_{\pm 0.5}$	90.8$_{\pm 0.2}$	85.0
	Uni-LoRA	0.023M	94.5$_{\pm 0.2}$	89.7$_{\pm 0.4}$	64.6$_{\pm 1.0}$	92.3$_{\pm 0.4}$	79.8$_{\pm 1.1}$	90.9$_{\pm 0.2}$	85.3
	PROLoSA	0.023M	94.9$_{\pm 0.3}$	89.5$_{\pm 0.6}$	63.5$_{\pm 0.2}$	92.5$_{\pm 0.2}$	81.7$_{\pm 0.6}$	90.8$_{\pm 0.2}$	85.5
LARGE	LoRA	0.786M	96.2$_{\pm 0.5}$	90.2$_{\pm 1.0}$	68.2$_{\pm 1.9}$	**94.8**$_{\pm 0.3}$	85.2$_{\pm 1.1}$	**92.3**$_{\pm 0.5}$	87.8
	VeRA	0.061M	96.1$_{\pm 0.1}$	90.9$_{\pm 0.7}$	68.0$_{\pm 0.8}$	94.4$_{\pm 0.2}$	85.9$_{\pm 0.7}$	91.7$_{\pm 0.8}$	87.8
	Tied-LoRA	0.066M	94.8$_{\pm 0.6}$	89.7$_{\pm 1.0}$	64.7$_{\pm 1.2}$	94.1$_{\pm 0.1}$	81.2$_{\pm 0.1}$	90.8$_{\pm 0.3}$	85.9
	VB-LoRA	0.162M†	96.1$_{\pm 0.2}$	91.4$_{\pm 0.6}$	68.3$_{\pm 0.7}$	94.7$_{\pm 0.5}$	86.6$_{\pm 1.3}$	91.8$_{\pm 0.1}$	88.2
	LoRA-XS	0.025M	95.9$_{\pm 0.3}$	90.7$_{\pm 0.4}$	67.0$_{\pm 1.2}$	93.9$_{\pm 0.1}$	**88.1**$_{\pm 0.3}$	92.0$_{\pm 0.1}$	87.9
	FourierFT	0.048M	96.0$_{\pm 0.2}$	90.9$_{\pm 0.3}$	67.1$_{\pm 1.4}$	94.4$_{\pm 0.4}$	87.4$_{\pm 1.6}$	92.0$_{\pm 0.4}$	88.0
	Uni-LoRA	0.023M	96.3$_{\pm 0.2}$	91.3$_{\pm 0.6}$	68.5$_{\pm 1.1}$	94.6$_{\pm 0.4}$	86.6$_{\pm 1.6}$	92.1$_{\pm 0.1}$	88.3
	PROLoSA	0.023M	**96.5**$_{\pm 0.2}$	**91.7**$_{\pm 0.6}$	**70.3**$_{\pm 0.6}$	94.6$_{\pm 0.1}$	87.4$_{\pm 0.6}$	91.9$_{\pm 0.2}$	**88.7**

Method	Trainable Params	GSM8K	MATH
LoRA	200M	74.90	**31.28**
VeRA	1.90M	74.98	28.84
FourierFT	0.59M	72.97	25.14
Uni-LoRA	0.52M	**75.59**	28.94
ProLoSA	0.52M	74.69	29.90

Table 2: Results on mathematical reasoning benchmarks, including GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021).

Method	**# Params**	**Score$_1$**	**Score$_2$**
w/o FT	-	1.31 / 1.46	1.11 / 1.06
LoRA$^{\#}$	159.9M / 250.3M	**5.62** / 6.20	3.23 / 4.13
Uni-LoRA	0.52M / 1.0M	5.34 / 6.34	3.56 / 4.43
ProLoSA	0.52M / 1.0M	5.36 / **6.57**	**4.25** / **4.72**

Table 3: Instruction-tuning results under the QLoRA training recipe (Dettmers et al., 2023). Each entry reports LLaMA2-7B / LLaMA2-13B results. Score$_1$ and Score$_2$ denote MT-Bench single-turn and multi-turn scores, respectively.

Study	Variant	d	K	CoLA	MRPC
Budget	Compressed-only	23040	0	68.99$_{\pm 0.58}$	90.93$_{\pm 0.00}$
	Hybrid, small K	22320	720	69.62$_{\pm 1.84}$	**91.69**$_{\pm 0.40}$
	Hybrid, large K	14400	8640	**70.27**$_{\pm 0.60}$	90.77$_{\pm 0.42}$
	Sparse-only	0	23040	59.99$_{\pm 1.40}$	84.89$_{\pm 0.83}$
Support	Random	22320	720	66.58$_{\pm 0.63}$	90.28$_{\pm 0.61}$
	Accum. gradient	22320	720	67.95$_{\pm 0.59}$	90.03$_{\pm 0.31}$
	Warmup SNIP	22320	720	**69.62**$_{\pm 1.84}$	**91.69**$_{\pm 0.40}$
Warmup	0%	22320	720	66.58$_{\pm 0.63}$	90.28$_{\pm 0.61}$
	5%	22320	720	69.05$_{\pm 0.51}$	91.12$_{\pm 0.70}$
	10%	22320	720	**69.62**$_{\pm 1.84}$	**91.69**$_{\pm 0.40}$
	20%	22320	720	67.49$_{\pm 0.96}$	90.60$_{\pm 0.81}$

Table 4: Ablation results on CoLA and MRPC under matched trainable-parameter budgets. The budget study compares compressed-only, sparse-only, and representative hybrid allocations. The support and warmup studies fix $d = 22320$ and $K = 720$ and vary only the corresponding design choice.

5 Conclusion

We proposed ProLoSA, a compressed LoRA framework that augments a projected low-dimensional branch with a routed sparse residual adapter. The projected branch preserves the variance-reduction benefit of global sharing, while the sparse branch restores coordinate-wise flexibility for directions poorly represented by the compressed subspace. Our bias–variance analysis shows when compression can improve generalization and how sparse residual correction reduces projection bias. Experiments on synthetic validation, GLUE, mathematical reasoning, and instruction tuning demonstrate that ProLoSA achieves a strong performance–parameter trade-off, suggesting that effective PEFT should structure compressed parameters as carefully as it reduces their number.

Limitations

Despite the strong performance-parameter trade-off achieved by ProLoSA, several limitations in our current framework warrant further investigation. Primarily, our method relies on a static sparse support identified via data-dependent sensitivity scores

during an initial warmup phase. While this fixed routing ensures training stability without introducing substantial computational overhead, it may become suboptimal if the ideal residual directions shift significantly during the later stages of joint optimization. Exploring lightweight dynamic routing mechanisms could potentially address this rigidity. Furthermore, the empirical success of ProLoSA currently depends on manually tuning the budget allocation between the compressed latent dimension and the sparse residual capacity. Since the optimal ratio often varies across different downstream tasks, developing an automatic, task-adaptive allocation strategy would significantly improve the out-of-the-box usability of our approach. Finally, while our bias-variance analysis provides a principled explanation for the benefits of compressed adaptation, it is grounded in a local quadratic approximation of the population risk. Although this standard statistical formulation successfully predicts empirical behavior in our synthetic settings, fully bridging the theoretical gap between these approximations and the highly non-convex loss landscapes of large language models remains an open challenge.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 7319–7328.

Alan Ansell, Ivan Vulić, Hannah Sterz, Anna Korhonen, and Edoardo M. Ponti. 2024. Scaling sparse fine-tuning to large language models. *Preprint*, arXiv:2401.16405.

Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. 2025. LoRA-XS: Low-rank adaptation with extremely small number of parameters. In *28th European Conference on Artificial Intelligence*. IOS Press.

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

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36.

Gemma Team. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2 edition. Springer.

Shwai He, Liang Ding, Daize Dong, Jeremy Zhang, and Dacheng Tao. 2022. SparseAdapter: An easy approach for improving the parameter-efficiency of adapters. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2184–2190.

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. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. 2024. VeRA: Vector-based random matrix adaptation. In *International Conference on Learning Representations*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2019. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*.

Kaiyang Li, Shaobo Han, Qing Su, Wei Li, Zhipeng Cai, and Shihao Ji. 2025. Uni-LoRA: One vector is all you need. In *The 39th Conference on Neural Information Processing Systems*.

Yang Li, Shaobo Han, and Shihao Ji. 2024. VB-LoRA: Extreme parameter efficient fine-tuning with vector banks. *Advances in Neural Information Processing Systems*, 37:16724–16751.

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. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR.

Mahdi Nikdan, Soroush Tabesh, Elvir Crnčević, and Dan Alistarh. 2024. RoSA: Accurate parameter-efficient fine-tuning via robust adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR.

Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. 2024. Tied-LoRA: Enhancing parameter efficiency of LoRA with weight tying. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 8694–8705.

Weixi Song, Zuchao Li, Lefei Zhang, Hai Zhao, and Bo Du. 2024. Sparse is enough in fine-tuning pre-trained large language models. In *Proceedings of the 41st International Conference on Machine Learning*.

Aad W. van der Vaart. 1998. *Asymptotic Statistics*. Cambridge University Press.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley.

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 *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

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. In *International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36.

A Additional Theoretical Details

A.1 General Bias–Variance Decomposition with Finite-Sample Bias

The main text presents the locally centered case for clarity. Here we provide the more general decomposition that explicitly includes finite-sample estimation bias, optimization bias, or deviations from the ideal local quadratic model.

For classical LoRA, consider the local decomposition

$$\hat{\theta}_L = \theta^\star + b_L + \xi_L, \qquad \mathbb{E}[\xi_L] = 0, \qquad \mathrm{Cov}(\xi_L) = \Sigma_L, \tag{30}$$

where

$$b_L = \mathbb{E}[\hat{\theta}_L] - \theta^\star. \tag{31}$$

Substituting Eq. (30) into the local quadratic approximation in Eq. (4) gives

$$\mathbb{E}[R(\hat{\theta}_L)] - R(\theta^\star) \approx \frac{1}{2} b_L^\top H b_L + \frac{1}{2}\mathrm{tr}(H\Sigma_L). \tag{32}$$

When $b_L \approx 0$, this reduces to Eq. (5) in the main text.

For the compressed parameterization, let $\theta_U^\star$ be the H-weighted projection of $\theta^\star$ onto $\mathcal{S}_U = \mathrm{col}(P)$. Consider the decomposition

$$\hat{\theta}_U = \theta_U^\star + P b_z + P\zeta, \qquad \mathbb{E}[\zeta] = 0, \qquad \mathrm{Cov}(\zeta) = \Sigma_z, \tag{33}$$

where $P b_z \in \mathcal{S}_U$ captures finite-sample or optimization bias within the compressed subspace. Because $\theta_U^\star$ is the H-weighted projection of $\theta^\star$ onto $\mathcal{S}_U$, the projection residual is H-orthogonal to every direction in $\mathcal{S}_U$:

$$(\theta_U^\star - \theta^\star)^\top H P b_z = 0. \tag{34}$$

Thus,

$$\mathbb{E}[R(\hat{\theta}_U)] - R(\theta^\star) \approx \frac{1}{2}(\theta_U^\star - \theta^\star)^\top H(\theta_U^\star - \theta^\star) + \frac{1}{2} b_z^\top P^\top H P b_z + \frac{1}{2}\mathrm{tr}(H P \Sigma_z P^\top). \tag{35}$$

Therefore, in the presence of finite-sample bias, a sufficient condition for compressed LoRA to improve over classical LoRA is

$$(\theta_U^\star - \theta^\star)^\top H(\theta_U^\star - \theta^\star) + b_z^\top P^\top H P b_z + \mathrm{tr}(H P \Sigma_z P^\top) < b_L^\top H b_L + \mathrm{tr}(H\Sigma_L). \tag{36}$$

The simplified condition in Eq. (9) follows when $b_L \approx 0$ and $b_z \approx 0$. The main intuition remains unchanged: compression is beneficial when its variance reduction outweighs its projection bias and any additional compressed-space estimation bias.

A.2 Proof of Projection Bias Reduction

We provide the formal proof of Proposition 1. The argument applies to any fixed sparse support I and does not depend on how the support is selected.

Recall that compressed LoRA restricts the full LoRA parameter vector to

$$\mathcal{S}_U = \mathrm{col}(P) \subseteq \mathbb{R}^D. \tag{37}$$

Let $\theta^\star$ be the full-space population optimum and let $H \succeq 0$ be the local curvature around $\theta^\star$. The compressed-only local optimum is

$$\theta_U^\star = \arg\min_{\theta \in \mathcal{S}_U} (\theta - \theta^\star)^\top H(\theta - \theta^\star), \tag{38}$$

with approximation bias

$$\mathsf{Bias}_U = \frac{1}{2}(\theta_U^\star - \theta^\star)^\top H(\theta_U^\star - \theta^\star). \tag{39}$$

For ProLoSA, let $I = \{i_1, \ldots, i_K\} \subseteq \{1, \ldots, D\}$ denote the selected sparse support and let

$$E_I = [e_{i_1}, \ldots, e_{i_K}] \in \mathbb{R}^{D \times K}. \tag{40}$$

For this fixed support, ProLoSA optimizes over the augmented space

$$\mathcal{S}_{U,I} = \{Pz + E_I \alpha \mid z \in \mathbb{R}^d, \alpha \in \mathbb{R}^K\} = \mathrm{col}([P, E_I]). \tag{41}$$

The corresponding local optimum is

$$\theta^\star_{\mathcal{S}_{U,I}} = \arg\min_{\theta \in \mathcal{S}_{U,I}} (\theta - \theta^\star)^\top H (\theta - \theta^\star), \tag{42}$$

and its approximation bias is

$$\mathsf{Bias}_{U,I} = \frac{1}{2} (\theta^\star_{\mathcal{S}_{U,I}} - \theta^\star)^\top H (\theta^\star_{\mathcal{S}_{U,I}} - \theta^\star). \tag{43}$$

Since $\mathcal{S}_U = \mathrm{col}(P)$ and $\mathcal{S}_{U,I} = \mathrm{col}([P, E_I])$, we have

$$\mathcal{S}_U \subseteq \mathcal{S}_{U,I}. \tag{44}$$

The compressed-only optimum minimizes the local quadratic approximation error over the smaller feasible set $\mathcal{S}_U$, whereas the ProLoSA optimum minimizes the same objective over the larger feasible set $\mathcal{S}_{U,I}$. Therefore,

$$(\theta^\star_{\mathcal{S}_{U,I}} - \theta^\star)^\top H (\theta^\star_{\mathcal{S}_{U,I}} - \theta^\star) \leq (\theta^\star_U - \theta^\star)^\top H (\theta^\star_U - \theta^\star). \tag{45}$$

Multiplying both sides by $1/2$ gives

$$\mathsf{Bias}_{U,I} \leq \mathsf{Bias}_U. \tag{46}$$

This proves Proposition 1.

This result concerns only the best attainable local approximation bias. It does not imply that ProLoSA always has lower overall expected excess risk, because the sparse residual branch can introduce additional estimation variance. The full bias–variance condition under which ProLoSA improves over compressed-only LoRA is given in Appendix A.4.

Finally, the amount of bias reduction depends on the selected support. For any fixed support, adding sparse coordinate directions enlarges the feasible space and cannot increase the best attainable local approximation bias. Support selection determines how much of the compressed projection error can be recovered under a fixed sparse budget K.

A.3 Explicit Bias Reduction in the Isotropic Case

We further characterize how much projection bias can be removed by the sparse residual branch in a simplified isotropic setting. Assume that the local curvature is isotropic, $H = I$, and that the projection matrix has orthonormal columns, $P^\top P = I_d$. Let

$$Q_P = I - PP^\top \tag{47}$$

be the orthogonal projector onto the complement of $\mathrm{col}(P)$. The component of the full-space optimum that cannot be represented by the compressed branch is

$$q = Q_P \theta^\star. \tag{48}$$

Thus, the compressed-only projection bias is

$$\mathsf{Bias}_U = \frac{1}{2} \|q\|_2^2. \tag{49}$$

For ProLoSA, the augmented space includes both the projected subspace and the selected sparse coordinate directions. Since any component of $E_I\alpha$ inside $\mathrm{col}(P)$ can already be represented by the projected branch, only its orthogonal component contributes to reducing the off-subspace residual. Define

$$A_I = Q_P E_I. \tag{50}$$

The best sparse correction in the orthogonal complement solves

$$\min_{\alpha \in \mathbb{R}^K} \|q - A_I\alpha\|_2^2. \tag{51}$$

Let Π_{A_I} denote the orthogonal projector onto $\mathrm{col}(A_I)$. The residual off-subspace error after adding the best sparse correction is $(I - \Pi_{A_I})q$. Therefore,

$$\mathsf{Bias}_{U,I} = \frac{1}{2}\|(I - \Pi_{A_I})q\|_2^2. \tag{52}$$

Equivalently,

$$\mathsf{Bias}_{U,I} = \mathsf{Bias}_U - \frac{1}{2}\|\Pi_{A_I} q\|_2^2. \tag{53}$$

Thus, in this isotropic setting, the sparse residual branch removes exactly the portion of the compressed projection error that lies in the span of the selected sparse directions after removing their projected-subspace components.

Define the recovered fraction as

$$\rho_I^2 = \frac{\|\Pi_{A_I} q\|_2^2}{\|q\|_2^2}. \tag{54}$$

Then $\rho_I^2 \in [0, 1]$, and

$$\mathsf{Bias}_{U,I} = (1 - \rho_I^2)\mathsf{Bias}_U. \tag{55}$$

This expression clarifies when sparse residual correction is most effective. If the off-subspace error is concentrated on a small number of coordinates, a limited sparse support can recover a large fraction of the missing component. If the error is diffuse across many coordinates, the same sparse budget may provide only limited bias reduction.

A.4 Bias–Variance Condition for ProLoSA

The previous sections show that the sparse residual branch cannot increase the best attainable local approximation bias. However, adding K trainable sparse coefficients can increase estimation variance. This section gives the corresponding sufficient condition under which the bias reduction leads to lower expected excess risk.

Let

$$M_I = [P, E_I] \in \mathbb{R}^{D\times(d+K)} \tag{56}$$

denote the augmented reconstruction matrix for a fixed support I. Assume that the empirical ProLoSA estimator can be locally written as

$$\hat{\theta}_I = \theta^\star_{\mathcal{S}_{U,I}} + M_I\eta, \qquad \mathbb{E}[\eta] = 0, \qquad \mathrm{Cov}(\eta) = \Sigma_I. \tag{57}$$

Using the local quadratic approximation in Eq. (4), its expected excess risk is

$$\mathbb{E}[R(\hat{\theta}_I)] - R(\theta^\star) \approx \mathsf{Bias}_{U,I} + \frac{1}{2}\mathrm{tr}(HM_I\Sigma_I M_I^\top). \tag{58}$$

The corresponding compressed-only estimator satisfies

$$\mathbb{E}[R(\hat{\theta}_U)] - R(\theta^\star) \approx \mathsf{Bias}_U + \frac{1}{2}\mathrm{tr}(HP\Sigma_z P^\top). \tag{59}$$

Therefore, ProLoSA improves over compressed-only LoRA when

$$\mathsf{Bias}_{U,I} + \frac{1}{2}\mathrm{tr}(HM_I\Sigma_I M_I^\top) < \mathsf{Bias}_U + \frac{1}{2}\mathrm{tr}(HP\Sigma_z P^\top). \tag{60}$$

Equivalently,

$$\mathsf{Bias}_U - \mathsf{Bias}_{U,I} > \frac{1}{2}\left[\mathrm{tr}(HM_I\Sigma_I M_I^\top) - \mathrm{tr}(HP\Sigma_z P^\top)\right]. \tag{61}$$

This condition formalizes the trade-off behind ProLoSA. The sparse residual branch is beneficial when the projection bias it recovers is larger than the additional estimation variance introduced by the sparse coefficients. Thus, Proposition 1 should be interpreted as a bias monotonicity result, while Eq. (61) gives the risk-level condition.

A.5 Connection to Warmup SNIP Support Selection

The analysis above holds for any fixed sparse support I. In practice, ProLoSA selects this support using sensitivity scores accumulated during a compressed warmup stage. During warmup, the sparse branch is inactive and the reconstructed LoRA parameter vector is

$$\theta_D^{(t)} = P\theta_d^{(t)}. \tag{62}$$

Consider perturbing coordinate i of $\theta_D^{(t)}$. A first-order Taylor approximation of the mini-batch loss gives

$$\begin{aligned} \mathcal{L}^{(t)}(\theta_D^{(t)} + \delta_i e_i) - \mathcal{L}^{(t)}(\theta_D^{(t)}) \approx g_i^{(t)}\delta_i, \\ g^{(t)} = \nabla_{\theta_D^{(t)}}\mathcal{L}^{(t)}. \end{aligned} \tag{63}$$

Following SNIP-style saliency criteria, we measure the sensitivity of coordinate i by the first-order effect of removing or perturbing its current value. Setting $\delta_i = -\theta_{D,i}^{(t)}$ gives

$$\left|\theta_{D,i}^{(t)} g_i^{(t)}\right|. \tag{64}$$

Accumulating this quantity over the warmup stage yields

$$s_i = \sum_{t=1}^{T_{\mathrm{w}}} \left|\theta_{D,i}^{(t)} g_i^{(t)}\right|. \tag{65}$$

Coordinates with large s_i are those whose perturbation has a consistently large first-order effect on the training loss during compressed adaptation. Selecting the top-K coordinates therefore provides a data-dependent heuristic for identifying directions where the compressed branch may require local correction. This support-selection rule does not guarantee the globally optimal support in the theoretical sense, but it aligns the sparse residual budget with coordinates that are locally influential under the compressed warmup trajectory.

A.6 Training Algorithm

Algorithm 1 summarizes the full training procedure of ProLoSA. The first stage performs compressed warmup and accumulates coordinate-wise sensitivity scores. The second stage fixes the selected sparse support and jointly optimizes the projected and sparse residual branches.

Algorithm 1 ProLoSA with Warmup SNIP Support Selection

Require: Frozen backbone W_0; projection matrix P; sparse budget K; warmup steps T_{w}; total training steps T.

Initialize compressed latent parameter θ_d.
Set sparse adapter inactive and initialize score vector $s \leftarrow 0_D$.
for $t = 1$ to T_{w} **do**
  Reconstruct $\theta_D^{(t)} \leftarrow P\theta_d^{(t)}$ and unpack it into $\{A_\ell, B_\ell\}_{\ell \in \mathcal{L}}$.
  Compute mini-batch loss $\mathcal{L}^{(t)}$ and gradient $g^{(t)} \leftarrow \nabla_{\theta_D^{(t)}} \mathcal{L}^{(t)}$.
  Accumulate scores $s \leftarrow s + \left|\theta_D^{(t)} \odot g^{(t)}\right|$.
  Update θ_d by back-propagation.
end for
Select sparse support $I \leftarrow \mathrm{TopK}(s, K)$.
Construct $R = [e_{i_1}, \ldots, e_{i_K}]$ for $I = \{i_1, \ldots, i_K\}$ and initialize $\alpha \leftarrow 0$.
for $t = T_{\mathrm{w}} + 1$ to T **do**
  Reconstruct $\theta_D \leftarrow P\theta_d + R\alpha$ and unpack it into $\{A_\ell, B_\ell\}_{\ell \in \mathcal{L}}$.
  Compute mini-batch loss $\mathcal{L}^{(t)}$.
  Update θ_d and α by back-propagation.
end for
return Trained θ_d, α, and fixed routing matrix R.

B Additional Experimental Details

This appendix provides implementation details omitted from the main text for reproducibility. Unless otherwise specified, the pretrained backbone is frozen and only PEFT parameters are updated. For each benchmark, all compared methods use the same data split, target modules, optimizer, batch size, number of epochs, learning-rate schedule, and random seeds whenever applicable. For **ProLoSA**, the sparse support is selected after compressed warmup using the accumulated sensitivity score defined in Section 3. The selected support is fixed during the remaining training stage, and the sparse coefficients are initialized to zero. All reported training time and memory statistics include method-specific stages, including the warmup stage of **ProLoSA**.

B.1 Synthetic Experiment Configuration

Table 5 summarizes the configuration used for the synthetic validation experiment in Section 4.2.

Hyperparameter	Value
Full dimension D	512
Compressed dimension d	32
Sparse budget K	4
Feature covariance	I_D
Noise std. σ_y	$\{0.25, 0.5, 1.0\}$
Mismatch range γ	$\{0, 0.5, 1.0, 1.5, 2.0\}$
Sample sizes n	$\{64, 128, 256\}$
Number of trials M	100
Ridge coefficient λ	10^{-6}
Projection type	normalized one-hot
Sparse support	warmup SNIP score
Evaluation metric	population excess risk
Random seed	13

Table 5: Configuration of the synthetic bias–variance validation experiment.

B.2 GLUE Hyperparameters

Table 6 reports the main hyperparameters for GLUE. We use the same settings for all compared PEFT methods under the corresponding backbone unless otherwise stated. For CoLA, STS-B, and the remaining tasks, we report Matthews correlation, Pearson correlation, and accuracy, respectively. The median performance over five random seeds is reported.

Backbone	Task	d	K	Epochs	BS	LR
RoBERTa-base	SST-2	22320	720	60	32	5e-4
RoBERTa-base	MRPC	22320	720	30	32	1e-3
RoBERTa-base	CoLA	14400	8640	80	32	2e-2
RoBERTa-base	QNLI	22320	720	25	32	1e-3
RoBERTa-base	RTE	22320	720	160	32	1e-2
RoBERTa-base	STS-B	22320	720	80	32	2e-4
RoBERTa-large	SST-2	22320	720	20	32	1e-3
RoBERTa-large	MRPC	22320	720	40	32	2e-4
RoBERTa-large	CoLA	14400	8640	40	32	5e-3
RoBERTa-large	QNLI	22320	720	20	32	5e-4
RoBERTa-large	RTE	22320	720	40	32	5e-3
RoBERTa-large	STS-B	22320	720	40	32	2e-4
Optimizer			AdamW			
Scheduler			Linear			
Seeds			5			
ProLoSA warmup ratio			10%			

Table 6: Core hyperparameter settings for GLUE experiments. BS denotes batch size. The d and K columns denote the compressed latent dimension and sparse residual budget used by ProLoSA.

B.3 Mathematical Reasoning Hyperparameters

Table 7 reports the core configuration for GSM8K and MATH. All methods use the same backbone, target modules, quantization setting, and decoding protocol. Exact-match scores are computed with the same answer extraction rule for all methods.

Hyperparameter	**GSM8K**	**MATH**
Shared Training Configuration		
Backbone	Gemma-7B	
Train data	MetaMathQA train	
Epochs	2	
Train batch size	1	
Gradient accumulation	64	
Learning rate	2×10^{-4}	
Scheduler	Cosine	
Max sequence length	512	
ProLoSA d	349525	
ProLoSA K	174763	
ProLoSA warmup ratio	10%	
Seeds	42	
Task-Specific Evaluation Configuration		
Metric	Exact match	
Eval batch size	32	32
Max new tokens	1024	2048

Table 7: Core training and evaluation settings for mathematical reasoning experiments. The model is fine-tuned once on MetaMathQA and evaluated across both benchmarks.

B.4 Instruction-Tuning Hyperparameters

Table 8 reports the core configuration for instruction tuning. All methods are trained on the same instruction data and evaluated with the same MT-Bench protocol. For QLoRA-style experiments, the backbone is quantized and frozen, and only PEFT parameters are updated.

B.5 Ablation Details

Table 9 reports the full budget-allocation sweep between the projected branch and the sparse residual branch. All variants use the same total trainable-parameter budget $B = d + K = 23040$. The compressed-only variant corresponds to $K = 0$, the sparse-only variant corresponds to $d = 0$, and intermediate configurations correspond to **ProLoSA** with both branches active. Since B is fixed, increasing the sparse budget K necessarily reduces the compressed latent dimension d.

Hyperparameter	Shared / LLaMA2-7B	LLaMA2-13B
Training data	Cleaned Alpaca	
Evaluation	MT-Bench	
Quantization	4-bit NF4 + double quantization	
Epochs	1	
Effective batch size	16	
Per-device batch size	4	2
Gradient accumulation	4	8
Scheduler	Linear	
Max sequence length	16 / 512	
Decoding strategy	Category-specific sampling/greedy	
Temperature	0.7 / 0.1 / 0 by category	
Max new tokens	1024	
Evaluator / judge	GPT-4 single	
LoRA params	159.9M	250.3M
Uni-LoRA params	0.52M	1.0M
ProLoSA params	0.52M	1.0M
ProLoSA d	507904	983040
ProLoSA K	16384	65536
ProLoSA warmup ratio	10%	
Seeds	0	

Table 8: Core training and evaluation settings for instruction-tuning experiments. Score_1 and Score_2 denote MT-Bench single-turn and multi-turn scores, respectively.

Study	Variant	d	K	CoLA	MRPC
Budget	Compressed-only	23040	0	$68.99_{\pm 0.58}$	$90.93_{\pm 0.00}$
	ProLoSA	22680	360	$68.29_{\pm 1.40}$	$90.11_{\pm 0.50}$
	ProLoSA	22320	720	$69.62_{\pm 1.84}$	$\mathbf{91.69}_{\pm 0.40}$
	ProLoSA	21600	1440	$67.21_{\pm 0.84}$	$90.87_{\pm 0.46}$
	ProLoSA	20160	2880	$68.72_{\pm 0.45}$	$83.74_{\pm 9.65}$
	ProLoSA	18720	4320	$67.51_{\pm 0.51}$	$90.03_{\pm 0.42}$
	ProLoSA	14400	8640	$\mathbf{70.27}_{\pm 0.60}$	$90.77_{\pm 0.42}$
	ProLoSA	5760	17280	$68.75_{\pm 1.27}$	$89.71_{\pm 0.20}$
	Sparse-only	0	23040	$59.99_{\pm 1.40}$	$84.89_{\pm 0.83}$
Support	Random	22320	720	$66.58_{\pm 0.63}$	$90.28_{\pm 0.61}$
	Accum. gradient	22320	720	$67.95_{\pm 0.59}$	$90.03_{\pm 0.31}$
	Warmup SNIP	22320	720	$\mathbf{69.62}_{\pm 1.84}$	$\mathbf{91.69}_{\pm 0.40}$
Warmup	0%	22320	720	$66.58_{\pm 0.63}$	$90.28_{\pm 0.61}$
	5%	22320	720	$69.05_{\pm 0.51}$	$91.12_{\pm 0.70}$
	10%	22320	720	$\mathbf{69.62}_{\pm 1.84}$	$\mathbf{91.69}_{\pm 0.40}$
	20%	22320	720	$67.49_{\pm 0.96}$	$90.60_{\pm 0.81}$

Table 9: Ablation details on budget allocation, sparse support selection, and warmup length. CoLA reports Matthews correlation and MRPC reports accuracy.