RSLoRA: Dynamic Rank Allocation via Representational Similarity

Anonymous authors
Paper under double-blind review

Abstract

Low-Rank Adaptation (LoRA) freezes the pretrained parameters and introduces additional trainable low-rank modules. Determining an appropriate rank for each LoRA module remains a nontrivial problem. Prior approaches typically assign module-wise ranks by estimating module importance using information derived from the module itself or from interactions among modules, but largely neglect the relationship between each module and its associated pretrained weights. As noted by Hu et al. (2021), LoRA modules effectively amplify task-relevant directions that are already present in the pretrained model but are insufficiently emphasized. Hence, rank allocation should more appropriately be guided by characterizing the relationship between each module and its associated pretrained weights. Motivated by this observation, we propose RSLoRA, which introduces a new evaluation mechanism for rank allocation based on representational similarity. Specifically, RSLoRA measures the similarity between representations produced by the LoRA branch and the pretrained branch using Centered Kernel Alignment (CKA). Under a fixed rank budget, we devise a rank allocation criterion based on the principle that modules exhibiting greater representational dissimilarity (i.e., lower CKA) should be assigned larger rank capacity, as they must model structures that are not adequately captured by the pretrained weights. Overall, our method offers a representation-centric perspective on rank allocation. Experiments across multiple pretrained language models and downstream tasks demonstrate that RSLoRA consistently outperforms state-of-the-art parameter-efficient fine-tuning (PEFT) baselines.

1 Introduction

Low-Rank Adaptation (LoRA) (Hu et al., 2021) has become a widely adopted parameter-efficient fine-tuning (PEFT) method (Wang et al., 2025; Han et al., 2024) for adapting pretrained models to downstream tasks. LoRA freezes each pretrained weight matrix and parameterizes its task-specific update using two trainable low-rank factors. The rank directly determines computational cost while constraining the adaptation capacity of each module (Hu et al., 2021; Zeng & Lee, 2024), making appropriate rank configuration a central challenge in LoRA fine-tuning.

Standard LoRA (Hu et al., 2021) adopts a static and uniform rank configuration, ignoring the heterogeneous adaptation needs of different layers and modules (Zhang et al., 2023; Lu et al., 2024; Voita et al., 2019). Under a fixed rank budget, this design can lead to a suboptimal allocation where critical modules are under-provisioned, while less important ones receive redundant capacity (Deng et al., 2026). To address this limitation, recent studies have explored adaptive rank allocation strategies by computing a heuristic importance score, which is derived from parameter sensitivity, sparsity, or the intrinsic structure of LoRA updates, to guide heterogeneous rank assignments (Zhang et al., 2023; Liu et al., 2026; Ke et al., 2024). Despite their different criteria, these methods primarily characterize LoRA modules from the adaptation side, without explicitly incorporating their relationship with the corresponding pretrained weights into rank allocation.

Previous studies have shown that LoRA updates are closely related to their corresponding pretrained weights. Hu et al. (2021) showed that LoRA tends to amplify task-relevant directions that already exist but are not emphasized in the pretrained model. Furthermore, NoRM (Jiang et al., 2025) revealed that repeatedly amplifying already-emphasized directions is closely associated with parame-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ter redundancy and fine-tuning performance. Since rank limits the number of independent directions available to LoRA for task-specific adaptation, these observations motivate us to consider the relationship between a LoRA module and its corresponding pretrained weights as a potential signal for rank allocation.

Motivated by the preceding discussion, we consider rank allocation from the perspective of the relationship between each LoRA update and its corresponding pretrained weights. Since rank allocation is task-specific (Deng et al., 2026; Lin et al., 2026), we study this relationship through the representations they induce on downstream-task inputs. We employ Centered Kernel Alignment (CKA) (Kornblith et al., 2019) to measure their representational relationship, as CKA captures the similarity between the sample-relation structures induced by two representations.

Based on this perspective, we propose **RSLoRA**, a dynamic rank allocation method that introduces the pretrained–LoRA representational relationship to adjust module ranks under a fixed global rank budget. The basic idea is that stronger representational dissimilarity indicates that the LoRA update captures structures less aligned with its corresponding pretrained weight, and such modules are therefore assigned relatively more rank to provide greater flexibility for learning task-specific adaptation directions. During fine-tuning, RSLoRA further considers the temporal variation of this dissimilarity and periodically adjusts module ranks under the fixed global rank budget.

Our main contributions are summarized as follows.

- We introduce a representational perspective for adaptive rank allocation. Rather than characterizing LoRA modules solely through adaptation-side information, we consider how each LoRA update relates to its corresponding pretrained weights on downstream-task inputs, providing a task-conditioned signal for module-wise rank allocation.
- We propose RSLoRA, a dynamic rank allocation method that combines pretrained–LoRA representational dissimilarity with its temporal variation to guide rank adjustment. Under a fixed global rank budget, RSLoRA performs bidirectional rank adjustment through sensitivity-based pruning and zero-impact expansion, allowing modules to dynamically release or regain rank throughout fine-tuning.
- Extensive experiments demonstrate that RSLoRA consistently outperforms strong PEFT baselines under comparable parameter budgets, while further analyses validate the effectiveness of its representational allocation criterion and dynamic rank adjustment.

2 Related Work

2.1 Adaptive Rank Allocation for LoRA

LoRA (Hu et al., 2021) parameterizes the weight updates with low-rank factors, where the rank governs both the adaptation capacity and the training efficiency of each module. However, LoRA assigns a uniform rank to all modules, overlooking their heterogeneous adaptation needs and leading to suboptimal performance under a fixed rank budget (Voita et al., 2019; Zhang et al., 2023; Ding et al., 2023). To overcome this limitation, recent studies have explored adaptive rank allocation.

Existing approaches fall into two paradigms. **(1) Importance-based rank allocation** assigns ranks according to heuristic scores. Based on the scoring criteria, they can be further divided into two categories. *Optimization-state-based methods* (Zhang et al., 2023; Cui et al., 2026; Zhang et al., 2025; Zhao et al., 2026) derive such scores from optimization dynamics, including gradient–weight products, sensitivity measures, and integrated gradients. *Matrix-structure-based methods* (Mao et al., 2024; Liu et al., 2026; Kumaravelu et al., 2026) exploit properties of LoRA updates, such as parameter magnitudes, singular values, and spectral statistics. **(2) Direct rank learning or selection methods** (Ding et al., 2023; Zhang et al., 2024; Shinwari & Usama, 2025; Valipour et al., 2023) learn or search for rank allocations during training instead of explicitly defining heuristic scores. Despite these advances, existing allocation criteria are largely derived from the adaptation parameters themselves. Consequently, the interaction between task-specific adaptation and the pretrained model remains underexplored as a source of information for rank allocation.

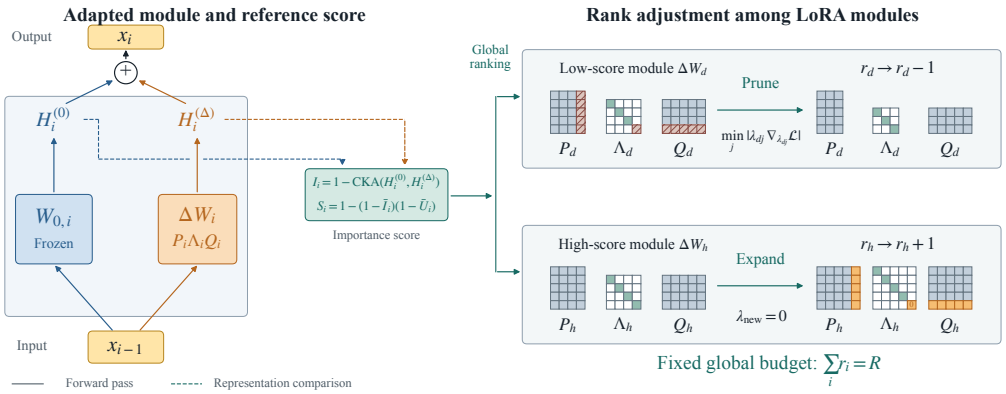


Figure 1: Overview of RSLoRA. RSLoRA measures pretrained–LoRA representational dissimilarity and its temporal variation to guide rank allocation. Under a fixed global rank budget, modules with lower importance scores remove the least sensitive rank components, while those with higher scores receive additional rank.

2.2 Representational Similarity and Centered Kernel Alignment

Representational similarity analysis characterizes relationships between internal representations that may not be apparent from model parameters or final performance alone. Representative measures include SVCCA (Raghu et al., 2017), PWCCA (Morcos et al., 2018), and Centered Kernel Alignment (CKA) (Kornblith et al., 2019), with CKA characterizing similarity through the sample-relation structures induced by two representations. CKA has been widely used to analyze the representational structure of neural networks, such as cross-layer similarity in fine-tuned Transformers (Phang et al., 2021) and critical layers in pretrained models (Liu et al., 2025). Beyond post-hoc analysis, representational similarity has also been used to guide structural decisions, including layer pruning (Ren et al., 2023; Hu et al., 2024; Pons et al., 2024) and layer selection for LoRA fine-tuning (Ando Ogawa et al., 2026). These studies highlight that representational similarity not only provides a tool for analyzing internal representations but can also serve as an informative signal for model adaptation and structural decisions.

3 Method

We propose **RSLoRA**, a dynamic rank allocation framework based on the pretrained–LoRA representational relationship to adjust module ranks under a fixed global rank budget. RSLoRA consists of two main components: (i) an importance score that combines pretrained–LoRA representational dissimilarity with its temporal variation, and (ii) a bidirectional rank adjustment strategy that dynamically reallocates rank across modules according to the resulting importance scores.

3.1 Preliminaries

Low-Rank Adaptation. For the pretrained weight matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, LoRA (Hu et al., 2021) parameterizes its update as

$$W = W_0 + \Delta W = W_0 + BA, \qquad A \in \mathbb{R}^{r \times d_{\text{in}}}, \quad B \in \mathbb{R}^{d_{\text{out}} \times r}, \tag{1}$$

where $r \ll \min(d_{\text{in}}, d_{\text{out}})$ is a hyperparameter that denotes the rank of the LoRA update. For each adapted module, we refer to the paths parameterized by the frozen pretrained weight matrix and the low-rank update as the pretrained branch and the LoRA branch, respectively. Standard LoRA typically uses a uniform rank across adapted modules, which may limit its flexibility in adaptation.

To enable adaptive rank allocation, prior work (Zhang et al., 2023; Liu et al., 2026) commonly adopts an SVD-inspired parameterization of the low-rank update:

$$W = W_0 + \Delta W = W_0 + P\Lambda Q, \tag{2}$$

where $P \in \mathbb{R}^{d_{\text{out}} \times r}$, $Q \in \mathbb{R}^{r \times d_{\text{in}}}$, and $\Lambda = \text{Diag}(\lambda_1, \lambda_2, \ldots, \lambda_r) \in \mathbb{R}^{r \times r}$. Following prior work, orthogonality regularization is applied to encourage different rank components to capture mutually orthogonal adaptation directions:

$$R(P, Q) = \|P^\top P - I\|_F^2 + \|QQ^\top - I\|_F^2. \tag{3}$$

Under this parameterization, the low-rank update can be decomposed as

$$\Delta W = \sum_{j=1}^{r} \lambda_j \boldsymbol{p}_{:,j} \boldsymbol{q}_{j,:}, \tag{4}$$

where each triplet $(\boldsymbol{p}_{:,j}, \lambda_j, \boldsymbol{q}_{j,:})$ defines a rank-one component. The parameterization enables the module rank to be adjusted by operating on individual rank components during training. Following these adaptive LoRA methods, we adopt the same parameterization for dynamic rank adjustment.

Centered Kernel Alignment (CKA) (Kornblith et al., 2019) measures the similarity between neural network representations by comparing the sample-relation structures they induce. Given the same m input samples, let $Y \in \mathbb{R}^{m \times d_y}$ and $Z \in \mathbb{R}^{m \times d_z}$ denote two representation matrices. Their Gram matrices are defined as

$$K = YY^\top, \qquad L = ZZ^\top, \tag{5}$$

where each entry encodes pairwise relations among samples in the corresponding representation spaces. Let $C = I_m - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$ denote the centering matrix. The Hilbert–Schmidt Independence Criterion (HSIC) (Gretton et al., 2005) between K and L is

$$\text{HSIC}(K, L) = \frac{1}{(m-1)^2} \operatorname{tr}(KCLC), \tag{6}$$

where $\operatorname{tr}(\cdot)$ denotes the matrix trace. CKA is then defined as

$$\text{CKA}(Y, Z) = \frac{\text{HSIC}(K, L)}{\sqrt{\text{HSIC}(K, K)\,\text{HSIC}(L, L)}}. \tag{7}$$

By comparing the centered Gram matrices, CKA measures representational similarity based on the agreement between the sample-relation structures induced by the two representations. A larger CKA value indicates greater similarity between the sample-relation structures induced by the two representations.

3.2 Importance Score via Representational Similarity

From a representational perspective, a key question is how to translate the pretrained–LoRA representational relationship into a criterion for rank allocation. Our key idea is to characterize this relationship through representational similarity and use it to explore the relative demand for adaptation directions across LoRA modules. Stronger representational similarity suggests a lower relative demand for adaptation directions and thus greater potential for rank reduction, whereas lower similarity motivates preserving relatively more rank to retain additional directions for representing structures different from those of the pretrained branch.

To quantify the pretrained–LoRA representational relationship, we use CKA, which compares representations without requiring dimension-wise feature correspondence (Kornblith et al., 2019), to measure the similarity of the representational structures induced by the pretrained weight and the LoRA update on the same inputs. At training step t, given the same module input $X_i^{(t)}$, the representations induced by the two branches are

$$H_i^{(0,t)} = X_i^{(t)} W_{0,i}^\top, \qquad H_i^{(\Delta,t)} = X_i^{(t)} \Delta W_i^{(t)\top}. \tag{8}$$

We compute these representations on a fixed calibration subset $\mathcal{D}_c$. A calibration-set sensitivity analysis is provided in Appendix H.1. Accordingly, we define the pretrained–LoRA representational dissimilarity of module i as

$$I_i^{(t)} = 1 - \text{CKA}\left(H_i^{(0,t)}, H_i^{(\Delta,t)}\right). \tag{9}$$

A larger $I_i^{(t)}$ indicates greater representational dissimilarity between the LoRA branch and its pretrained counterpart. Since a single estimate of $I_i^{(t)}$ may be affected by the finite calibration samples and stochastic optimization, we smooth it using an exponential moving average:

$$\bar{I}_i^{(t)} = \beta_1 \bar{I}_i^{(t-\tau)} + (1-\beta_1) I_i^{(t)}, \tag{10}$$

While $\bar{I}_i^{(t)}$ reflects the smoothed level of representational dissimilarity, its temporal variation indicates whether the relative demand for adaptation directions remains stable during fine-tuning. Large variation suggests that this demand is still evolving, motivating the preservation of sufficient adaptation directions to avoid premature rank reduction and retain flexibility for subsequent adaptation. We quantify this temporal variation by introducing $\bar{U}_i^{(t)}$:

$$\bar{U}_i^{(t)} = \beta_2 \bar{U}_i^{(t-\tau)} + (1-\beta_2) \left| I_i^{(t)} - \bar{I}_i^{(t)} \right|, \tag{11}$$

where $0 < \beta_1, \beta_2 < 1$. Here, $\bar{I}_i^{(t)}$ denotes the smoothed representational dissimilarity, while $\bar{U}_i^{(t)}$ captures its recent temporal variation.

For rank allocation, we assign lower importance scores to LoRA modules that remain consistently similar to their pretrained counterparts. Accordingly, we define

$$S_i^{(t)} = \bar{I}_i^{(t)} + \bar{U}_i^{(t)} - \bar{I}_i^{(t)} \bar{U}_i^{(t)} = 1 - \left(1 - \bar{I}_i^{(t)}\right) \left(1 - \bar{U}_i^{(t)}\right), \tag{12}$$

This formulation yields a low score when representational dissimilarity is both small and stable, while either large dissimilarity or substantial temporal variation keeps the score high. Alternative aggregation strategies are evaluated in Section 4.4.

3.3 Global Rank Allocation Strategy

RSLoRA uses $S_i^{(t)}$ to dynamically adjust module ranks under a fixed global rank budget. The training process consists of three stages: warm-up, rank allocation, and stabilization. The warm-up stage establishes initial adaptation directions before rank adjustment begins, the rank allocation stage dynamically adjusts module ranks according to their importance scores, and the stabilization stage fixes the resulting rank configuration for final optimization. The complete training procedure is summarized in Algorithm 1.

Warm-up stage. The global rank budget is initially distributed uniformly across the n LoRA modules, with $r_i = R/n$. The model is first optimized under this fixed rank configuration, allowing initial adaptation directions to form before the pretrained–LoRA representational relationship is used to guide rank adjustment.

Rank allocation stage. As the pretrained–LoRA representational relationship continues to evolve during fine-tuning, the relative importance of modules may change over time. RSLoRA therefore periodically adjusts module ranks according to their current importance scores while keeping the global rank budget fixed. Every τ training steps, RSLoRA computes $\{S_i^{(t)}\}_{i=1}^n$ for all modules and ranks them globally. At each adjustment event, the bottom-$b(t)$ modules each decrease their rank by one, while the top-$b(t)$ modules each increase their rank by one. The schedule $b(t)$ is detailed in Appendix C.

For each selected low-scoring module i, RSLoRA removes the rank component with the smallest first-order sensitivity, $\rho_{i,j}^{(t)} = \left| \lambda_{i,j}^{(t)} \frac{\partial \mathcal{L}}{\partial \lambda_{i,j}^{(t)}} \right|$, thereby reducing the immediate effect of rank reduction on the training objective. For each selected high-scoring module, RSLoRA adds a new rank component with its diagonal coefficient initialized to zero, enabling rank expansion without immediately perturbing the current optimization state.

Stabilization stage. RSLoRA stops rank adjustment and continues optimizing the model under the final rank configuration until convergence.

Algorithm 1 RSLoRA

Require: Training set $\mathcal{D}$, calibration set $\mathcal{D}_c$, total iterations T, allocation interval τ, allocation period $[t_{\mathrm{w}}, t_{\mathrm{f}}]$, fixed rank budget R, rank schedule $b(t)$, EMA coefficients β_1, β_2

Initialize $\{P_i, \Lambda_i, Q_i\}_{i=1}^{n}$ with $\sum_{i=1}^{n} r_i = R$
for $t = 1, \ldots, T$ **do**
  Sample a mini-batch from $\mathcal{D}$ and update LoRA parameters
  if $t_{\mathrm{w}} < t \leq t_{\mathrm{f}}$ and $(t - t_{\mathrm{w}}) \bmod \tau = 0$ **then**
    for all $i = 1, \ldots, n$ **do**
      Compute $H_i^{(0,t)}$ and $H_i^{(\Delta,t)}$ on $\mathcal{D}_c$
      Compute representational dissimilarity $I_i^{(t)}$ as (9)
      Update $\bar{I}_i^{(t)}$ as (10) and $\bar{U}_i^{(t)}$ as (11)
    end for
    Compute importance score $\boldsymbol{S_i^{(t)}}$ by (12)
    Prune one least-important active rank from each of the bottom-$b(t)$ modules using $\rho_{i,j}^{(t)}$
    Activate one zero-impact rank in each of the top-$b(t)$ modules
  end if
end for
return Fine-tuned LoRA parameters

4 Experiments

4.1 Experimental Settings

Computational Resources. We implement all methods in PyTorch (Paszke et al., 2019) using the Hugging Face Transformers (Wolf et al., 2020) and PEFT libraries and conduct all experiments on NVIDIA A100 GPUs running Ubuntu 22.04.

Models and Tasks. We evaluate RSLoRA on natural language understanding (NLU) and commonsense reasoning (CR). We use DeBERTaV3-base (He et al., 2021) for NLU and LLaMA-3-8B (Grattafiori et al., 2024) for CR.

Datasets. For NLU, we evaluate on the GLUE benchmark (Wang et al., 2018), which covers a diverse range of tasks, including linguistic acceptability, sentiment analysis, semantic similarity, paraphrase identification, and natural language inference. For CR, we consider eight benchmarks: BoolQ, PIQA, OpenBookQA, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, and SocialIQA (Sap et al., 2019).

Baselines. We compare RSLoRA with representative baselines: (i) Full fine-tuning method. (ii) standard LoRA with a fixed rank (Hu et al., 2021). (iii) AdaLoRA (Zhang et al., 2023). All baseline methods are implemented using publicly available codebases. Additional experimental details are provided in Appendix D. Unless otherwise specified, hyperparameters follow recommended settings in prior work to ensure fair comparison.

4.2 Natural Language Understanding

Table 1 reports results on eight GLUE tasks using DeBERTaV3-base. RSLoRA achieves the highest average score of 89.6. Under a comparable parameter budget, RSLoRA improves the average score over AdaLoRA and FlexLoRA from 88.8 and 88.3 to 89.6, respectively. The improvements are observed across different types of language understanding tasks. For example, RSLoRA achieves 71.2 on CoLA and 88.5 on RTE, showing strong performance on both linguistic acceptability and textual entailment. Overall, these results demonstrate that RSLoRA provides a more effective rank allocation strategy under a constrained parameter budget across diverse NLU tasks.

4.3 Commonsense Reasoning

Table 2 reports the results on eight commonsense reasoning benchmarks using LLaMA-3-8B. At $r = 8$, RSLoRA obtains an average accuracy of 87.2, compared with 87.1 for both AdaLoRA

Table 1: Results on the GLUE benchmark with DeBERTaV3-base. All results are averaged over 5 runs with different random seeds. Detailed results are provided in Appendix H.5.

Method	Param.	MNLI Acc.	SST-2 Acc.	CoLA Mcc.	QQP Acc.-F1	QNLI Acc.	RTE Acc.	MRPC Acc.-F1	STS-B Corr.	Avg.
Full FT	184M	89.9	95.6	69.2	**92.4**	94.0	83.8	90.2	91.6	88.3
BitFit	0.1M	89.7	95.7	67.8	88.3	93.2	84.2	89.7	91.6	87.5
H-Adapter	1.2M	89.5	94.7	68.6	90.7	94.1	85.4	91.1	90.5	88.0
P-Adapter	1.2M	89.4	94.8	68.7	90.5	93.8	84.4	90.0	91.4	87.9
LoRA$_{r=8}$	1.3M	89.4	93.5	67.5	89.6	94.0	86.5	91.5	91.4	87.9
AdaLoRA$_{r=8}$	1.9M	90.4	95.9	69.2	90.6	94.3	86.8	91.7	**91.7**	88.8
FlexLoRA$_{r=8}$	1.9M	89.5	95.4	68.1	90.6	94.1	86.2	90.8	91.4	88.3
DyLoRA$_{r=8}$	0.9M	89.9	95.5	67.7	86.1	94.2	84.5	92.2	87.7	87.2
DoRA	1.3M	88.5	93.2	67.6	90.6	94.0	78.9	91.3	90.1	86.8
RSLoRA$_{r=8}$	1.9M	**90.7**	**96.4**	**71.2**	90.8	**94.6**	**88.5**	**92.7**	**91.7**	**89.6**

and FlexLoRA and 82.6 for uniform-rank LoRA. Under this budget, RSLoRA remains competitive with adaptive-rank baselines and achieves particularly strong results on ARC-Challenge and OpenBookQA, indicating that its allocation strategy remains effective across different reasoning formats. When the average rank increases to 32, RSLoRA further improves to 88.0, exhibiting a clearer advantage over both LoRA and AdaLoRA. By contrast, AdaLoRA reaches 86.8 and LoRA reaches 84.5. Notably, LoRA at $r = 32$ remains below RSLoRA at $r = 8$, indicating that effective capacity allocation is more important than simply increasing the rank budget. The larger improvement at $r = 32$ further suggests that RSLoRA can make effective use of additional capacity by selectively assigning it to modules with higher estimated adaptation demand.

Table 2: Results on commonsense reasoning tasks.

Method	Param.	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
Full FT	8B	75.3	89.9	81.5	95.8	87.6	91.6	79.3	87.4	86.1
LoRA$_{r=8}$	14.2M	72.6	83.5	79.7	94.0	85.2	87.5	75.6	83.0	82.6
AdaLoRA$_{r=8}$	21.2M	75.6	88.7	82.7	**96.5**	88.7	93.0	82.8	88.6	87.1
RSLoRA$_{r=8}$	21.2M	74.5	89.8	81.7	94.3	89.3	93.6	83.5	**90.6**	87.2
LoRA$_{r=32}$	56.6M	71.3	86.4	79.0	94.4	86.9	90.2	80.9	87.0	84.5
AdaLoRA$_{r=32}$	56.6M	73.2	89.7	82.5	95.8	88.2	93.1	82.1	89.6	86.8
FlexLoRA	21.2M	74.6	89.3	82.6	96.2	88.4	93.0	82.6	89.8	87.1
LoRA-Dash	56.6M	75.2	88.5	81.7	96.2	88.0	92.6	81.6	87.4	86.4
NoRA+	56.6M	71.4	86.1	80.2	73.4	83.3	87.1	74.6	80.8	79.6
PrecLoRA	56.6M	76.0	89.9	82.8	93.8	87.2	93.5	83.3	87.4	86.7
IGULoRA	56.6M	54.1	84.0	70.0	84.3	65.2	89.2	74.4	71.6	74.1
RSLoRA$_{r=32}$	56.6M	**76.1**	**90.7**	**83.2**	**96.5**	**89.5**	**93.9**	**84.2**	90.0	**88.0**

4.4 Ablation Studies

Alternative importance metrics. We first examine whether pretrained–LoRA representational dissimilarity provides an effective signal for rank allocation. We compare RSLoRA with three adaptation-side alternatives: sensitivity, Frobenius norm, and spectral entropy. The sensitivity metric measures the average first-order influence of the retained rank components on the training objective, while the latter two characterize the magnitude and spectral distribution of the LoRA update, respectively. Formal definitions are provided in Appendix G.1. As shown in Table 3, RSLoRA delivers stronger overall performance on representative GLUE tasks than alternatives. These results support the use of the pretrained–LoRA representational relationship as a more reliable signal for rank allocation, beyond criteria derived solely from the adaptation parameters.

Representation similarity metric. We next examine whether the choice of representation similarity measure affects rank allocation. We replace linear CKA with centered cosine similarity, RBF-CKA, and RSA-Spearman while keeping the remaining allocation procedure unchanged. For each metric, the corresponding similarity is converted into representational dissimilarity. Formal definitions are provided in Appendix G.2. As shown in Table 3, the superior overall performance of RSLoRA supports its effectiveness for characterizing the pretrained–LoRA representational relationship used for rank allocation.

Table 3: Ablation of alternative allocation metrics of RSLoRA on three representative GLUE tasks.

Metric	CoLA Mcc.	RTE Acc.	SST-2 Acc.	Avg.
Sensitivity	69.7	88.4	95.4	84.5
Frobenius norm	67.6	86.6	95.5	83.2
Spectral entropy	69.8	86.6	95.4	83.9
Centered cosine	67.9	87.7	95.9	83.8
RBF-CKA	68.8	88.4	95.4	84.2
RSA-Spearman	68.1	86.6	94.7	83.1
RSLoRA	**71.2**	**88.5**	**96.4**	**85.4**

Table 4: Ablation of different variants of RSLoRA on three representative GLUE tasks.

Variant	CoLA Mcc.	RTE Acc.	SST-2 Acc.	Avg.
$S_i = \bar{I}_i$	67.9	84.4	95.5	82.6
$S_i = \bar{U}_i$	69.1	87.0	95.4	83.8
$S_i = (\bar{I}_i + \bar{U}_i)/2$	69.1	85.9	95.9	83.6
$S_i = \bar{I}_i \bar{U}_i$	68.1	87.0	95.4	83.5
Random	68.1	86.6	95.3	83.3
$\rho_{i,j} = \lvert\lambda_{i,j}\rvert$	68.8	87.0	95.3	83.7
RSLoRA	**71.2**	**88.5**	**96.4**	**85.4**

Variants of the importance score. We evaluate the design of the importance score by comparing the proposed aggregation with four alternatives: $S_i = \bar{I}_i$; $S_i = \bar{U}_i$; $S_i = (\bar{I}_i + \bar{U}_i)/2$; and $S_i = \bar{I}_i \bar{U}_i$. As shown in Table 4, the proposed aggregation achieves consistently higher performance than the alternative schemes. The results support the effectiveness of the proposed conservative aggregation strategy, which avoids assigning low importance to a module when either representational dissimilarity or temporal variation remains strong.

Rank pruning criteria. We further examine how rank components should be removed from low-scoring modules. Besides the first-order sensitivity used by RSLoRA, we consider two alternatives: (i) random removal and (ii) magnitude-based removal, which selects the component with the smallest $|\lambda_{i,j}|$. As shown in Table 4, both variants degrade performance because they may introduce undesirable perturbations during training. RSLoRA instead removes the least consequential directions, thereby reducing the disturbance caused by rank adjustment.

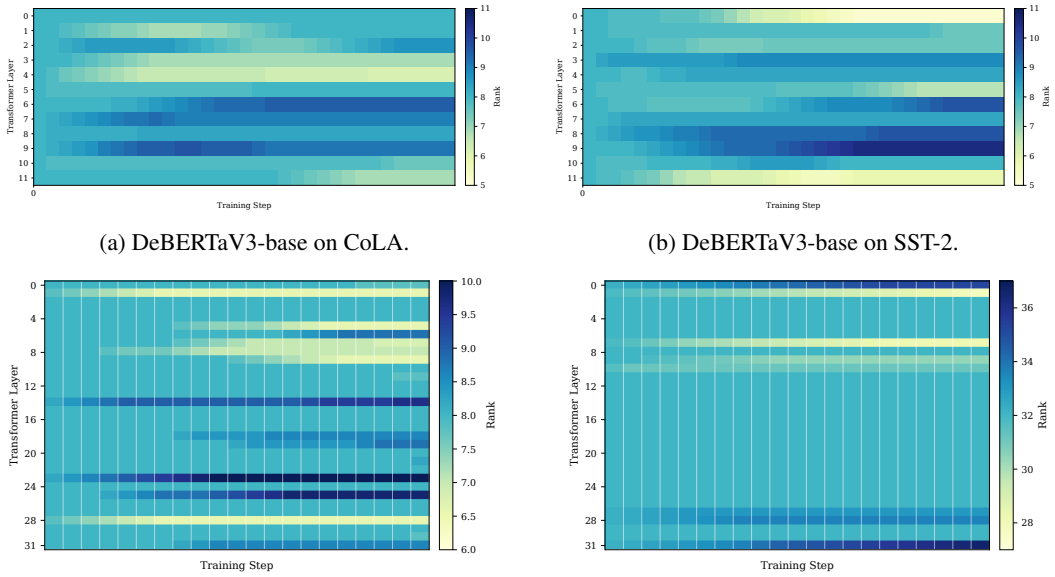
5 Analysis

5.1 Rank Allocation Dynamics across Tasks

Figures 2a and 2b show the evolution of rank allocation during fine-tuning. Starting from the uniform rank configuration, RSLoRA progressively develops heterogeneous and task-dependent rank distributions under a fixed global rank budget.

Both tasks allocate relatively large ranks to selected middle-to-upper modules, whereas several lower and topmost modules retain smaller ranks. From the representational perspective underlying RSLoRA, the larger ranks concentrated in selected middle-to-upper modules suggest that these modules play a greater role in transforming pretrained representations toward task-specific features. In contrast, the smaller ranks in several lower modules suggest greater preservation and reuse of general, transferable representations, indicating greater potential for rank compression. This interpretation is consistent with prior findings that lower layers tend to preserve more transferable representations, while fine-tuning often induces larger representational changes in higher layers (Hao et al., 2019; Merchant et al., 2020; Mosbach et al., 2020).

CoLA and SST-2 exhibit different degrees of rank heterogeneity. SST-2 produces a more polarized allocation, with adaptation capacity concentrated in a smaller subset of modules, whereas CoLA distributes capacity more uniformly. One possible explanation is that sentiment classification



(a) DeBERTaV3-base on CoLA.

(b) DeBERTaV3-base on SST-2.

(c) LLaMA-3-8B with $r = 8$.

(d) LLaMA-3-8B with $r = 32$.

Figure 2: Layer-wise rank allocation dynamics of RSLoRA. (a)–(b) show DeBERTaV3-base on CoLA and SST-2 at $r = 8$, and (c)–(d) show LLaMA-3-8B at $r = 8$ and $r = 32$, respectively. Darker colors indicate larger allocated ranks.

places greater emphasis on specialized transformations of sentiment-relevant semantic information, whereas linguistic acceptability judgment draws on a broader range of linguistic information.

5.2 Rank Allocation under Different Budgets

We visualize the evolution of layer-wise ranks during fine-tuning under average rank budgets of $r = 8$ and $r = 32$. As shown in Figures 2c and 2d, the two budgets lead to different rank distributions. At $r = 8$, rank adjustments occur broadly across layers, resulting in pronounced rank differentiation. In contrast, at $r = 32$, most layers remain close to the initial uniform rank, while substantial adjustments are concentrated in a smaller subset of layers.

Importantly, the relatively uniform rank distribution at $r = 32$ does not imply that adaptive rank allocation becomes unnecessary. Despite the more concentrated adjustments, RSLoRA achieves an average accuracy of 88.0, compared with 84.5 for uniform-rank LoRA under the same budget. This observation suggests that, when the overall rank capacity is less constrained, effective allocation may rely more on selectively adjusting the capacity of a subset of modules than on broadly differentiating ranks across the entire network.

6 Conclusion

We propose RSLoRA, a dynamic rank allocation framework that incorporates the pretrained–LoRA representational relationship into module-wise rank allocation. RSLoRA combines representational dissimilarity with its temporal variation to construct an importance score and performs bidirectional rank adjustment under a fixed global rank budget. Extensive experiments demonstrate the effectiveness of RSLoRA, with consistent improvements over strong PEFT baselines. Further analyses reveal distinct rank distributions across tasks and rank budgets, showing that RSLoRA adapts its rank distribution to different downstream tasks and budget constraints. Overall, these results demonstrate the effectiveness of incorporating the pretrained–LoRA representational relationship into adaptive rank allocation.

AI USE STATEMENT

Large language models (LLMs) were used as writing assistants during the preparation of this manuscript. Their use was limited to polishing the original text by correcting grammar, improving clarity and readability, and ensuring a consistent academic writing style. No part of the research design, methodology, experiments, result analysis, or scientific conclusions was generated by LLMs. All AI-assisted text was reviewed and verified by the authors, who take full responsibility for the content and integrity of the final manuscript. The use of LLMs did not constitute a substantive contribution that would merit authorship.

REPRODUCIBILITY STATEMENT

For clarity and reproducibility, we summarize the key methodological and experimental details provided in the main paper and appendices as follows. Codes are available at: https://anonymous.4open.science/r/RSLoRA1-1EA7

- **Algorithmic Details.** Section 3 provides the complete formulation of RSLoRA, including the representation-based importance score and global rank adjustment strategy. The complete training procedure is summarized in Algorithm 1.
- **Implementation Details.** The computation of CKA and the rank adjustment schedule are described in Appendices A and C, respectively.
- **Experimental Setup.** Section 4.1 describes the evaluated tasks, models, and baselines. Appendix D provides the dataset splits, evaluation metrics, training settings, and hyperparameters for experiments.
- **Ablation Protocols.** The definitions of the alternative allocation metrics are provided in Appendix G and the additional controlled ablations are documented in Appendix H.

REFERENCES

Sergio A Alvarez. Gaussian rbf centered kernel alignment (cka) in the large-bandwidth limit. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):6587–6593, 2022.

Keith Ando Ogawa, Bruno Lopes Yamamoto, Lucas Lauton de Alcantara, Lucas Pellicer, Rosimeire Pereira Costa, Edson Bollis, Anna Helena Reali Costa, and Artur Jordao. Layer-wise lora fine-tuning: A similarity metric approach. In *International Conference on Pattern Recognition*, pp. 370–385. Springer, 2026.

Xuan Cui, Huiyue Li, Run Zeng, Yunfei Zhao, Jinrui Qian, Wei Duan, Bo Liu, and Zhanpeng Zhou. Igu-lora: Adaptive rank allocation via integrated gradients and uncertainty-aware scoring. *arXiv preprint arXiv:2603.13792*, 2026.

Guanzhi Deng, Bo Li, Ronghao Chen, Huacan Wang, Linqi Song, and Lijie Wen. Dr-lora: Dynamic rank lora for mixture-of-experts adaptation. *arXiv e-prints*, pp. arXiv–2601, 2026.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 4133–4145, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pp. 63–77. Springer, 2005.

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

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Visualizing and understanding the effectiveness of bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4143–4152, 2019.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021.

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

Seungbeom Hu, ChanJun Park, Andrew Ferraiuolo, Sang-Ki Ko, Jinwoo Kim, Haein Song, and Jieung Kim. Mpruner: Optimizing neural network size with cka-based mutual information pruning. *arXiv preprint arXiv:2408.13482*, 2024.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 5254–5276, 2023.

Shuyang Jiang, Yusheng Liao, Ya Zhang, Yanfeng Wang, and Yu Wang. Fine-tuning with reserved majority for noise reduction. In *International conference on learning representations*, volume 2025, pp. 30495–30514, 2025.

Wenjun Ke, Jiahao Wang, Peng Wang, Jiajun Liu, Dong Nie, Guozheng Li, and Yining Li. Unveiling lora intrinsic ranks via salience analysis. *Advances in Neural Information Processing Systems*, 37: 131575–131595, 2024.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMLR, 2019.

Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:249, 2008.

Vishnuprasadh Kumaravelu, Sunil Gupta, and PK Srijith. Post-optimization adaptive rank allocation for lora. *arXiv preprint arXiv:2604.27796*, 2026.

Weicheng Lin, Yi Zhang, Jiawei Dang, and Liang-Jie Zhang. Tlora: Task-aware low rank adaptation of large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29252–29269, 2026.

Muqing Liu, Chongjie Si, and Yuheng Jia. Flexlora: Entropy-guided flexible low-rank adaptation. *arXiv preprint arXiv:2601.22905*, 2026.

Xuyuan Liu, Lei Hsiung, Yaoqing Yang, and Yujun Yan. Spectral insights into data-oblivious critical layers in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4860–4877, 2025.

Haodong Lu, Chongyang Zhao, Jason Xue, Lina Yao, Kristen Moore, and Dong Gong. Adaptive rank, reduced forgetting: Knowledge retention in continual learning vision-language models with dynamic rank-selective lora. 2024.

Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. Dora: Enhancing parameter-efficient fine-tuning with dynamic rank distribution. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11662–11675, 2024.

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. What happens to bert embeddings during fine-tuning? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 33–44, 2020.

Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *Advances in neural information processing systems*, 31, 2018.

Marius Mosbach, Anna Khokhlova, Michael A Hedderich, and Dietrich Klakow. On the interplay between fine-tuning and sentence-level probing for linguistic knowledge in pre-trained transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2502–2516, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Jason Phang, Haokun Liu, and Samuel R Bowman. Fine-tuned transformers show clusters of similar representations across layers. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 529–538, 2021.

Ian Pons, Bruno Yamamoto, Anna H Reali Costa, and Artur Jordao. Effective layer pruning through similarity metric perspective. In *International Conference on Pattern Recognition*, pp. 423–438. Springer, 2024.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

Xiaoning Ren, Yun Lin, Yinxing Xue, Ruofan Liu, Jun Sun, Zhiyong Feng, and Jin Song Dong. Deeparc: Modularizing neural networks for the model maintenance. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pp. 1008–1019. IEEE, 2023.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

Haseeb Ullah Khan Shinwari and Muhammad Usama. Ard-lora: Dynamic rank allocation for parameter-efficient fine-tuning of foundation models with heterogeneous adaptation needs. *IEEE Transactions on Artificial Intelligence*, 2025.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3274–3287, 2023.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 5797–5808, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 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*, pp. 353–355, 2018.

Luping Wang, Sheng Chen, Linnan Jiang, Shu Pan, Runze Cai, Sen Yang, and Fei Yang. Parameter-efficient fine-tuning in large language models: a survey of methodologies. *Artificial Intelligence Review*, 58(8):227, 2025.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Yuchen Zeng and Kangwook Lee. The expressive power of low-rank adaptation. In *International Conference on Learning Representations*, volume 2024, pp. 5078–5123, 2024.

Hao Zhang, Bo Huang, Zhenjia Li, Xi Xiao, Hui Yi Leong, Zumeng Zhang, Xinwei Long, Tianyang Wang, and Hao Xu. Sensitivity-lora: Low-load sensitivity-based fine-tuning for large language models. *arXiv preprint arXiv:2509.09119*, 2025.

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

Ruiyi Zhang, Rushi Qiang, Sai Ashish Somayajula, and Pengtao Xie. Autolora: Automatically tuning matrix ranks in low-rank adaptation based on meta learning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5048–5060, 2024.

Xing Zhao, Liu Yang, Xi-Le Zhao, Yueqi Yang, and Tianqi Jiang. Hebbian-guided bi-directional rank adaptation for parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 19112–19128, 2026.

A CKA Computation Protocol

RSLoRA evaluates linear CKA on a fixed calibration set sampled from the training data. At each rank adjustment event, identical task inputs are passed through the pretrained and LoRA branches of every adapted module, and CKA is computed over the representations induced by the two branches at aligned non-padding token positions. The calibration set is selected once per run and reused throughout fine-tuning, so temporal changes in the pretrained–LoRA representational relationship reflect the evolution of the model representations rather than changes in the observation samples. Validation and test data are not used for rank allocation.

B Design of the Importance Score

The importance score combines the smoothed dissimilarity $\bar{I}_i$ and its temporal variation $\bar{U}_i$:

$$S_i = \bar{I}_i + \bar{U}_i - \bar{I}_i\bar{U}_i = 1 - (1 - \bar{I}_i)(1 - \bar{U}_i). \tag{13}$$

This aggregation is designed to support the bidirectional rank redistribution of RSLoRA.

For $\bar{I}_i, \bar{U}_i \in [0, 1]$, the score satisfies

$$\max(\bar{I}_i, \bar{U}_i) \le S_i \le \min(1, \bar{I}_i + \bar{U}_i), \tag{14}$$

and is monotone in both arguments because

$$\frac{\partial S_i}{\partial \bar{I}_i} = 1 - \bar{U}_i \ge 0, \qquad \frac{\partial S_i}{\partial \bar{U}_i} = 1 - \bar{I}_i \ge 0. \tag{15}$$

These properties allow RSLoRA to preserve modules that either maintain substantial representational dissimilarity or exhibit strong temporal variation, even when the other factor is small. In contrast, the multiplicative aggregation used in AdaLoRA (Zhang et al., 2023) requires both factors to be large and may suppress such modules when either factor is small, making it less suited to our bidirectional rank adjustment. The aggregation ablation in Table 4 further supports this design against alternative aggregation strategies.

C Rank Adjustment Schedule

As described in Section 3.3, we introduce a rank adjustment scheduler for the maximum number of rank adjustments at each adjustment event. Specifically, $b^{(t)}$ is gradually decreased during the rank-adjustment stage while the global rank budget remains fixed. The detailed schedule is defined as follows:

$$b^{(t)} = \max\left(1, \left\lceil b^{(0)} \left(1 - \frac{t - t_{\mathrm{w}}}{t_{\mathrm{f}} - t_{\mathrm{w}}}\right) \right\rceil\right), \qquad t_{\mathrm{w}} < t \le t_{\mathrm{f}}, \tag{16}$$

where $b^{(0)}$ is the initial transfer budget, t_{w} is the rank warm-up boundary, and t_{f} is the final permissible allocation step. The schedule is evaluated only at adjustment events satisfying $(t - t_{\mathrm{w}}) \bmod \tau = 0$.

Table 5: Details of the GLUE datasets.

Dataset	Task	#Train	#Dev	#Test	#Label	Metric
			Single-Sentence Classification			
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr.
SST-2	Sentiment	67k	872	1.8k	2	Accuracy
			Similarity and Paraphrase			
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy/F1
QQP	Paraphrase	364k	40k	391k	2	Accuracy/F1
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman corr.
			Natural Language Inference			
MNLI	NLI	393k	20k	20k	3	Accuracy
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy

Table 6: Hyperparameter settings for RSLoRA on the NLU task.

	CoLA	MNLI	MRPC	RTE	QNLI	SST-2	STS-B	QQP
Learning rate	8×10^{-4}	5×10^{-4}	1×10^{-3}	1.2×10^{-3}	5×10^{-4}	8×10^{-4}	2.2×10^{-3}	8×10^{-4}
Epochs	20	12	30	50	5	20	20	5
Batch size	32	32	32	32	32	32	32	32
γ	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

D Experimental Details

Unless otherwise specified, all methods on the same task use the same data split, optimization settings, evaluation protocol, and global rank budget.

D.1 Natural Language Understanding

For natural language understanding, we evaluate on the eight tasks of the GLUE benchmark (Wang et al., 2018). CoLA evaluates grammatical acceptability, while SST-2 evaluates binary sentiment classification. MRPC and QQP test whether two sentences or questions express equivalent meanings, whereas STS-B measures their graded semantic similarity. MNLI evaluates three-way natural language inference across multiple domains. QNLI tests whether a sentence contains the information needed to answer a given question, and RTE evaluates binary textual entailment.

We use the official training split of each task for optimization and report results on the GLUE development set. The dataset statistics and evaluation measures are summarized in Table 5. For MRPC and QQP, the reported score is the average of accuracy and F1. For STS-B, it is the average of Pearson and Spearman correlations.

We fine-tune DeBERTaV3-base (He et al., 2021). For all methods, the global rank budget corresponds to an average rank of 8 across adapted modules, and the LoRA scaling factor is set to $\alpha = 16$. We use AdamW with linear learning-rate decay. For RSLoRA across all NLU tasks, let T denote the total number of optimizer updates. We use a common relative rank allocation window, with $t_{\mathrm{w}} = \lfloor 0.1T \rfloor$ and $t_{\mathrm{f}} = \lfloor 0.9T \rfloor$. Given a nominal target of K allocation events, we set $\tau = \lceil (t_{\mathrm{f}} - t_{\mathrm{w}})/K \rceil$. Task-specific settings are provided in Table 6. Except for their respective rank allocation mechanisms, all compared methods use the same task-level training configuration.

D.2 Commonsense reasoning

For commonsense reasoning, we follow the Commonsense170K protocol used in prior PEFT studies (Hu et al., 2023). The training portions of eight benchmarks are combined into a unified

Table 7: Details of commonsense reasoning benchmarks

Dataset	Task type	#Test	Metric
BoolQ	Binary question answering	3270	Accuracy
PIQA	Physical commonsense reasoning	1838	Accuracy
SIQA	Social commonsense reasoning	1954	Accuracy
HellaSwag	Contextual completion	10042	Accuracy
WinoGrande	Commonsense coreference	1267	Accuracy
ARC-Easy	Scientific question answering	2376	Accuracy
ARC-Challenge	Scientific question answering	1172	Accuracy
OpenBookQA	Scientific question answering	500	Accuracy

instruction-tuning corpus. BoolQ evaluates binary question answering, PIQA physical commonsense, SocialIQA reasoning about social interactions, HellaSwag contextual completion, WinoGrande commonsense coreference resolution, ARC-Easy and ARC-Challenge grade-school science reasoning at different difficulty levels, and OpenBookQA open-book science question answering.

Following the Commonsense170K protocol, the training portions of these tasks are combined into a unified corpus containing 170420 examples. We hold out 1024 examples for validation and use the remaining 169396 examples for optimization. The same split is used for all compared methods. Validation examples are used only for training monitoring and checkpoint selection, while RSLoRA calibration examples are drawn exclusively from the optimization split.

Final performance is evaluated separately on each benchmark. We report accuracy for every task and use their unweighted arithmetic mean as the overall score. The hyperparameter settings used for the commonsense reasoning experiments are reported in Table 8. Unless otherwise specified, all compared methods use the same optimization settings.

Table 8: Hyperparameter settings for RSLoRA on commonsense reasoning tasks.

Hyperparameter	Setting	Hyperparameter	Setting
Backbone	LLaMA-3-8B	Optimizer	AdamW
Average rank r	$8/32$	LoRA scaling α	$16/64$
Learning rate	3×10^{-4}	LR schedule	Linear
Batch size	16	Dropout	0.05
Warm-up Steps	100	τ	1000
Rank warm-up steps	5000	Final fixed-allocation steps	5000
Epochs	3	Target modules	Q, K, V, Up, Down

E Comparison of Training Cost

In this section, we report system-level efficiency metrics, including total training runtime, peak allocated GPU memory, estimated training FLOPs, and evaluation runtime. As shown in Table 9, the results across the NLU and commonsense reasoning benchmarks consistently show that RSLoRA has a computational cost comparable to AdaLoRA, while maintaining nearly identical training FLOPs and consistently lower evaluation runtime.

On the NLU tasks, RSLoRA matches AdaLoRA in peak GPU memory, while being 3.8% faster on CoLA and only 1.8% slower on RTE. On commonsense reasoning, RSLoRA reduces the training time by 10.2% relative to AdaLoRA and also achieves a lower evaluation runtime. Although RSLoRA requires additional GPU memory on LLaMA-3-8B, the results overall indicate that representation-guided rank allocation introduces limited computational overhead.

Table 9: Comparison of training cost on the NLU and commonsense reasoning tasks.

Benchmark	Metric	LoRA	AdaLoRA	RSLoRA
CoLA	Train runtime (s)	362.4	703.3	676.9
	GPU memory (GB)	2.62	2.66	2.63
	Training FLOPs	2.325×10^{15}	2.343×10^{15}	2.327×10^{15}
	Evaluation runtime (s)	0.83	1.32	0.91
RTE	Train runtime (s)	260.3	503.9	513.0
	GPU memory (GB)	10.55	10.60	10.58
	Training FLOPs	1.168×10^{16}	1.177×10^{16}	1.169×10^{16}
	Evaluation runtime (s)	0.32	0.47	0.34
LLaMA3-8B	Train runtime (h)	24.15	36.38	32.67
	GPU memory (GB)	15.62	15.96	19.47
	Training FLOPs	2.761×10^{18}	2.763×10^{18}	2.763×10^{18}
	Evaluation runtime (s)	1885.80	2172.39	1911.94

F Analysis of Pretrained–LoRA Representational Relationship

To further examine the pretrained–LoRA representational relationship used by RSLoRA, we analyze the distribution and training dynamics of representational dissimilarity across modules and downstream tasks. Following Section 3.2, for module i we compute

$$I_i = 1 - \mathrm{CKA}\left(H_i^{(0)}, H_i^{(\Delta)}\right), \tag{17}$$

where $H_i^{(0)}$ and $H_i^{(\Delta)}$ are the representations induced by the pretrained and LoRA branches under the same module inputs, respectively. A larger I_i indicates weaker structural alignment between the two representations.

We fine-tune DeBERTaV3-base (He et al., 2021) with LoRA ($r = 8$) on the GLUE benchmark (Wang et al., 2018) and examine representational dissimilarity across modules, tasks, and training stages. As shown in Figure 3, representational dissimilarity varies substantially across modules, and its distribution differs across downstream tasks. The training dynamics on CoLA further show that different modules exhibit distinct trajectories over the course of fine-tuning. These observations indicate that the pretrained–LoRA representational relationship is heterogeneous across modules and tasks and evolves during downstream adaptation.

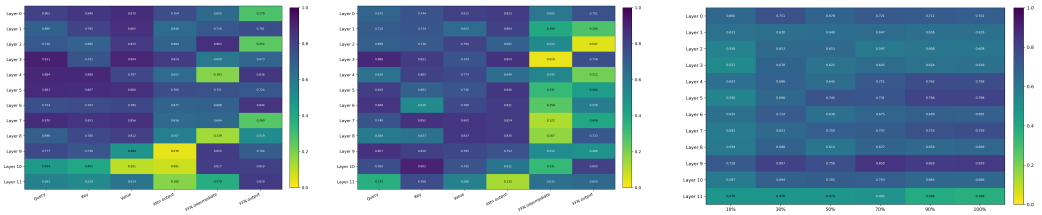
To examine whether this relationship reflects structure specific to the corresponding pretrained weights, we additionally construct a random-reference control. Specifically, we replace $W_{0,i}$ with a randomly initialized weight matrix while keeping the same module input X_i and compare the resulting representation with the LoRA branch. On CoLA, the mean CKA between the LoRA branch and the random-reference branch is 0.10, substantially lower than the mean CKA of 0.40 between the LoRA branch and its corresponding pretrained branch. This result suggests that the observed pretrained–LoRA relationship captures non-random structure associated with the corresponding pretrained weights, rather than arbitrary representational variation.

Together, these results provide empirical support for using the pretrained–LoRA representational relationship as a module-level signal for rank allocation.

G Definitions of Ablation Metrics

G.1 Importance Metrics

For the ablation study in Section 4.4, we consider three alternative module-importance metrics. Let $\mathcal{R}_i$ denote the set of retained rank components of module i, and let $\lambda_{i,j}$ denote the scaling factor associated with component $j \in \mathcal{R}_i$.



(a) SST-2 (b) CoLA (c) Layer-wise dynamics on CoLA

Figure 3: Pretrained–LoRA representational dissimilarity across modules, tasks, and training stages. (a)–(b) show module-level dissimilarity on SST-2 and CoLA, respectively, while (c) shows its evolution across Transformer layers during fine-tuning on CoLA. Dissimilarity is measured as $I_i = 1 - \mathrm{CKA}(H_i^{(0)}, H_i^{(\Delta)})$, with darker colors indicating greater dissimilarity.

Sensitivity. We measure the first-order sensitivity of each retained rank component to the training objective and average it within each module:

$$M_i^{\mathrm{sens}} = \frac{1}{|\mathcal{R}_i|} \sum_{j \in \mathcal{R}_i} \left| \lambda_{i,j} \frac{\partial \mathcal{L}}{\partial \lambda_{i,j}} \right|. \tag{18}$$

Frobenius norm. Let $H_i^{(\Delta,t)} \in \mathbb{R}^{n_i \times d_i}$ denote the representation produced by the LoRA branch of module i on the calibration inputs. We measure its normalized Frobenius norm:

$$M_i^{\mathrm{norm}} = \frac{\left\| H_i^{(\Delta,t)} \right\|_F}{\sqrt{n_i d_i}} = \left(\frac{1}{n_i d_i} \left\| H_i^{(\Delta,t)} \right\|_F^2 \right)^{1/2}. \tag{19}$$

Spectral entropy. We construct an energy distribution from the squared scaling factors of the retained rank components:

$$p_{i,j} = \frac{\lambda_{i,j}^2}{\sum_{k \in \mathcal{R}_i} \lambda_{i,k}^2}, \qquad j \in \mathcal{R}_i. \tag{20}$$

The normalized entropy is defined as

$$M_i^{\mathrm{ent}} = -\frac{1}{\log |\mathcal{R}_i|} \sum_{j \in \mathcal{R}_i} p_{i,j} \log\big(\max(p_{i,j}, \epsilon)\big). \tag{21}$$

For a module with only one retained component, or whose retained scaling factors are all zero, the entropy is defined as zero.

Each alternative metric is subsequently processed using the same temporal smoothing, score aggregation, and rank-adjustment procedure as RSLoRA.

G.2 Alternative Representational Similarity Measures

This section describes the alternative representational similarity measures used in the ablation study in Table 3. Each variant replaces linear CKA when comparing the pretrained representation $H_i^{(0)}$ with the LoRA representation $H_i^{(\Delta)}$, while leaving the remaining allocation procedure unchanged.

Centered cosine similarity. Let $\widetilde{H} = CH$ denote sample-centered features, where $C = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$. Centered cosine similarity is defined as

$$\mathrm{Cos}_c(H, G) = \frac{\langle \widetilde{H}, \widetilde{G} \rangle_F}{\|\widetilde{H}\|_F \|\widetilde{G}\|_F}. \tag{22}$$

This measure directly compares corresponding coordinates of the two centered representations.

Table 10: Sensitivity to the calibration set. S_1–S_5 correspond to calibration seeds 20260726–20260730, respectively; S_1 is the reference subset.

(a) Calibration-set size

Size	CoLA	MRPC	RTE	STS-B	Avg.
32	71.6	92.2	90.6	91.8	86.6
64	72.2	92.0	89.9	92.0	86.5
128	71.2	92.2	91.0	91.8	86.6
256	71.9	92.4	90.3	91.7	86.6
512	71.8	91.7	90.3	91.8	86.4

(b) Calibration-subset selection ($N = 512$)

Set	CoLA	MRPC	RTE	STS-B	Avg.	Δ
S_1	71.8	91.7	90.3	91.8	86.4	0.0
S_2	71.7	92.1	89.9	91.8	86.4	-0.1
S_3	72.2	92.2	90.6	91.7	86.7	+0.3
S_4	71.4	92.4	89.9	91.8	86.4	-0.1
S_5	71.0	92.2	89.8	91.9	86.2	-0.2

RBF-CKA. (Alvarez, 2022) We construct RBF kernels

$$K_{ab} = \exp\left(-\frac{\|h_a - h_b\|_2^2}{2\sigma_H^2}\right), \qquad L_{ab} = \exp\left(-\frac{\|g_a - g_b\|_2^2}{2\sigma_G^2}\right), \tag{23}$$

where each bandwidth is determined by the median of the positive pairwise squared distances in the corresponding representation. After centering the two kernel matrices, RBF-CKA is computed as

$$\mathrm{CKA}_{\mathrm{RBF}}(H, G) = \frac{\langle CKC, CLC\rangle_F}{\|CKC\|_F \|CLC\|_F}. \tag{24}$$

RSA-Spearman. (Kriegeskorte et al., 2008) We first center each representation and normalize every sample vector. For each branch, we then construct a sample-wise cosine-similarity matrix. Let v_H and v_G denote the vectorized upper-triangular entries of the two matrices. RSA-Spearman is defined as

$$\mathrm{RSA}_{\mathrm{Sp}}(H, G) = \rho_{\mathrm{Sp}}(v_H, v_G), \tag{25}$$

where ρ_{Sp} is Spearman's rank correlation. This measure compares the ordering of pairwise sample relations. Its value is likewise mapped from $[-1, 1]$ to $[0, 1]$.

For all variants, the normalized similarity $q_i^{(t)}$ is converted to representational dissimilarity as $I_i^{(t)} = 1 - q_i^{(t)}$ and is then processed by the same smoothing, score aggregation, and rank-adjustment procedure used by RSLoRA.

H Additional Ablation Studies

Unless otherwise specified, all variants use the same backbone, optimization settings, global rank budget, calibration protocol, and rank adjustment schedule as RSLoRA. Each experiment changes only the component under investigation.

H.1 Sensitivity to the Calibration Set

We evaluate the sensitivity of RSLoRA to the size and composition of the calibration set, as reported in Table 10.

Calibration-set size. The average score remains between 86.4 and 86.6 as the calibration-set size varies from 32 to 512, with no monotonic improvement from using more calibration examples. This indicates that RSLoRA can estimate its allocation signal from a relatively small calibration set.

Calibration-subset selection. With the calibration-set size fixed to 512, changing the sampled subset alters the average score by at most 0.3 points relative to the reference subset. The average performance remains stable despite some task-level variation, suggesting that RSLoRA has limited sensitivity to the specific composition of the calibration set.

H.2 Effect of Online Score Updating

To examine whether the allocation signal should be updated as training progresses, we compare RSLoRA with a frozen-score variant. This variant computes the module-wise importance scores at the first rank adjustment event and retains the resulting module ordering for all subsequent events. As shown in Table 11, the frozen-score variant obtains an average score of 83.0, compared with 85.4 for RSLoRA. The difference shows that refreshing the allocation signal is important for tracking changes in module-wise adaptation demand during optimization.

H.3 Representation-Space and Weight-Space Allocation Signals

The motivation of RSLoRA is to estimate adaptation demand from the input-conditioned relationship between the matched pretrained and LoRA branches. To examine whether this relationship can instead be measured directly in parameter space, we replace representation-space CKA with two weight-space alternatives. Weight-CKA applies linear CKA to the pretrained weight matrix and the corresponding LoRA update, whereas Weight-Cosine computes their centered cosine similarity. As shown in Table 11, Weight-CKA and Weight-Cosine obtain average scores of 82.6 and 84.1, respectively, compared with 85.4 for RSLoRA. The results suggest that representations provide a more informative allocation signal than weight space geometry alone. This finding is consistent with the motivation of RSLoRA: module-wise adaptation demand is better reflected by the functional relationship between the two branches than by their parameter-level similarity.

Table 11: Ablations of online score updating, representation space, and pretrained-reference correspondence on three GLUE tasks.

Variant	CoLA Mcc.	RTE Acc.	SST-2 Acc.	Avg.
Frozen-score	67.1	86.6	95.4	83.0
Weight-CKA	67.5	85.5	94.9	82.6
Weight-Cosine	69.7	87.3	95.4	84.1
Shuffled	68.8	87.3	95.8	84.0
RSLoRA	**71.2**	**88.5**	**96.4**	**85.4**

H.4 Matched Pretrained–LoRA Correspondence

To test the importance of pretrained–LoRA correspondence, we randomly shuffle the pretrained references across LoRA branches. As shown in Table 11, this modification reduces the average score from 85.4 to 84.0. This indicates that effective rank allocation depends on the relationship between each LoRA branch and its corresponding pretrained branch. Pretrained representations are not interchangeable references for rank allocation. Instead, the relationship between a LoRA branch and the specific pretrained branch it adapts provides module-specific information that is useful for determining rank allocation. More broadly, this finding provides a preliminary mechanistic insight into LoRA adaptation: the role of a LoRA update is not determined solely by the update itself, but is meaningfully related to the specific pretrained mapping it modifies. This pretrained-relative view offers a complementary perspective for understanding LoRA adaptation beyond analyzing the update in isolation.

H.5 Detailed Results on the GLUE Benchmark

Table 12 complements the main results in Table 1 by reporting the per-task means and standard deviations of the PEFT methods over five random seeds. These statistics provide additional information about the variability across runs. The average score is computed as the unweighted mean of the eight task scores.

Table 12: Results on the GLUE benchmark with DeBERTaV3-base. All results are reported as the mean and standard deviation over 5 runs with different random seeds.

Method	**Params.**	**MNLI** Acc.	**SST-2** Acc.	**CoLA** Mcc.	**QQP** Acc.-F1	**QNLI** Acc.	**RTE** Acc.	**MRPC** Acc.-F1	**STS-B** Corr.	**Avg.**
Full FT	184M	89.9	95.6	69.2	**92.4**	94.0	83.8	90.2	91.6	88.3
BitFit	0.1M	$89.7_{\pm 0.1}$	$95.7_{\pm 0.3}$	$67.8_{\pm 1.3}$	$88.3_{\pm 0.1}$	$93.2_{\pm 0.1}$	$84.2_{\pm 1.1}$	$89.7_{\pm 0.7}$	$91.6_{\pm 0.2}$	87.5
H-Adapter	1.2M	$89.5_{\pm 0.2}$	$94.7_{\pm 0.3}$	$68.6_{\pm 1.7}$	$90.7_{\pm 0.2}$	$94.1_{\pm 0.2}$	$85.4_{\pm 1.2}$	$91.1_{\pm 0.8}$	$90.5_{\pm 0.7}$	88.0
P-Adapter	1.2M	$89.4_{\pm 0.1}$	$94.8_{\pm 0.4}$	$68.7_{\pm 0.9}$	$90.5_{\pm 0.0}$	$93.8_{\pm 0.3}$	$84.4_{\pm 0.9}$	$90.0_{\pm 0.4}$	$91.4_{\pm 0.1}$	87.9
LoRA$_{r=8}$	1.3M	$89.4_{\pm 0.1}$	$93.5_{\pm 0.3}$	$67.5_{\pm 1.6}$	$89.6_{\pm 0.1}$	$94.0_{\pm 0.0}$	$86.5_{\pm 1.7}$	$91.5_{\pm 0.8}$	$91.4_{\pm 0.1}$	87.9
AdaLoRA$_{r=8}$	1.9M	$90.4_{\pm 0.1}$	$95.9_{\pm 0.2}$	$69.2_{\pm 1.6}$	$90.6_{\pm 0.1}$	$94.3_{\pm 0.1}$	$86.8_{\pm 1.1}$	$91.7_{\pm 0.3}$	$\mathbf{91.7}_{\pm 0.0}$	88.8
FlexLoRA$_{r=8}$	1.9M	$89.5_{\pm 0.1}$	$95.4_{\pm 0.1}$	$68.1_{\pm 0.8}$	$90.6_{\pm 0.1}$	$94.1_{\pm 0.3}$	$86.2_{\pm 1.5}$	$90.8_{\pm 0.6}$	$91.4_{\pm 0.1}$	88.3
DyLoRA$_{r=8}$	0.9M	$89.9_{\pm 0.7}$	$95.5_{\pm 0.2}$	$67.7_{\pm 2.0}$	$86.1_{\pm 0.2}$	$94.2_{\pm 0.3}$	$84.5_{\pm 0.9}$	$92.2_{\pm 0.7}$	$87.7_{\pm 1.5}$	87.2
DoRA	1.3M	$88.5_{\pm 0.1}$	$93.2_{\pm 0.1}$	$67.6_{\pm 0.6}$	$90.6_{\pm 0.1}$	$94.0_{\pm 0.1}$	$78.9_{\pm 2.0}$	$91.3_{\pm 1.3}$	$90.1_{\pm 0.4}$	86.8
RSLoRA$_{r=8}$	1.9M	$\mathbf{90.7}_{\pm 0.0}$	$\mathbf{96.4}_{\pm 0.1}$	$\mathbf{71.2}_{\pm 0.6}$	$90.8_{\pm 0.0}$	$\mathbf{94.6}_{\pm 0.2}$	$\mathbf{88.5}_{\pm 0.3}$	$\mathbf{92.7}_{\pm 0.2}$	$\mathbf{91.7}_{\pm 0.1}$	**89.6**