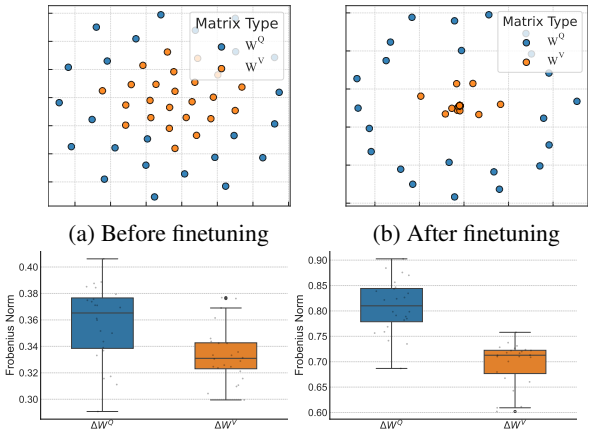
AILoRA: Function-Aware Asymmetric Initialization for Low-Rank Adaptation of Large Language Models

Anonymous ACL submission

Abstract

Parameter-efficient finetuning (PEFT) aims to mitigate the prohibitive computation and memory costs involved in adapting large-scale pretrained models to diverse downstream tasks. Among numerous PEFT strategies, Low-Rank Adaptation (LoRA) has emerged as one of the most widely adopted approaches due to its robust empirical performance and minimal implementation overhead. In practical deployment, LoRA is typically applied to the W^Q and W^V projection matrices of the attention modules, achieving an effective trade-off between model performance and parameter efficiency. While LoRA has achieved considerable success, it still encounters challenges such as suboptimal performance and slow convergence. To address these limitations, we introduce **AILoRA**, a novel parameter-efficient method that incorporates function-aware asymmetric low-rank priors. Our empirical analysis reveals that the projection matrices W^Q and W^V in the attention mechanism exhibit distinct parameter characteristics, stemming from their functional differences. Specifically, W^Q captures task-specific semantic space knowledge essential for attention distribution computation, making its parameters highly sensitive to downstream task variations. In contrast, W^V encodes token-level feature representations that tend to remain stable across tasks and layers. Leveraging these insights, AILoRA performs a function-aware initialization by injecting the principal components of W^Q to retain task-adaptive capacity, and the minor components of W^V to preserve generalizable feature representations. This asymmetric initialization strategy enables LoRA modules to better exploit the specialized roles of attention matrices, thereby enhancing both finetuning performance and convergence efficiency. Extensive experiments on multiple large language models and diverse natural language tasks demonstrate the consistent superiority of AILoRA over existing PEFT approaches.



(a) Before finetuning (b) After finetuning

(c) Box plot on CoLA dataset (d) Box plot on SST-2 dataset

Figure 1: Comparative analysis of the W^Q and W^V projection matrices in the self-attention mechanism. Figures (a) and (b) visualize the W^Q and W^V matrices across all layers of RoBERTa-large (24 encoder layers) before and after fine-tuning on the CoLA dataset, using t-SNE for dimensionality reduction. Each point represents a projection matrix from a specific layer. Figures (c) and (d) report the Frobenius norms of the weight updates ΔW^Q and ΔW^V after fine-tuning on the CoLA and SST-2 datasets, respectively.

1 Introduction

Large Language Models (Brown et al., 2020; Ouyang et al., 2022; Mao et al., 2024, LLMs), pretrained on large-scale text corpora, have exhibited remarkable generalization capabilities and broad applicability across a wide range of NLP tasks (Zheng et al., 2023), including question answering (Ivison et al., 2023), mathematical reasoning (Wang et al., 2024b) and tabular tasks (He et al., 2025). In practice, full finetuning is generally regarded as the most effective approach for adapting large language models to specific downstream tasks. However, the substantial computational and memory costs of full finetuning limit its applicability in real-world scenarios. For instance, finetuning a LLaMA-65B model requires over 780GB

of GPU memory (Dettmers et al., 2024), and mandates storing a full set of model parameters for each downstream task.

To address these challenges, parameter-efficient finetuning (Hu et al., 2023, PEFT) has emerged as an effective alternative to full finetuning for adapting large-scale pretrained models to downstream tasks, typically by freezing most model parameters and updating only a small number of trainable parameters or modules. Recents years have witnessed the rapid emergence of numerous PEFT methods, including Adapter tuning (Houlsby et al., 2019), Prefix-tuning (Li and Liang, 2021), LoRA (Hu et al., 2021), and BitFit (Zaken et al., 2022). Among these approaches, Low-Rank Adaptation (LoRA) has received particular attention due to its strong empirical performance and high parameter efficiency. Specifically, LoRA reduces finetuning overhead by decomposing the weight updates into the product of two low-rank matrices, which are typically applied to the W^Q and W^V matrices of the attention modules in practice to achieve a balance between parameter efficiency and model performance. Nevertheless, an increasing number of empirical studies have shown that the default initialization of LoRA often fails to yield optimal adaptation performance in downstream applications. With the aim of improving the initialization of LoRA modules, PiSSA (Meng et al., 2024) and MiLoRA (Wang et al., 2024a) utilize heuristically selected components from pretrained weights to initialize the low-rank matrices, aiming to enhance adaptation performance. However, these methods fail to take into account the distinct functional roles of the attention projection matrices W^Q and W^V, and a uniform singular value-based initialization may still be insufficient to achieve optimal performance on downstream tasks. This limitation becomes more pronounced as model size increases, where the oversimplified initialization strategy struggles to accommodate the growing complexity and functional heterogeneity of large-scale models.

To overcome previously described limitations and improve the initialization scheme in LoRA, we first examine the functional differences and parameter behaviors of the attention projection matrices W^Q and W^V. Prior studies (Vaswani et al., 2017; Clark et al., 2019) have shown that the W^Q projection matrices in attention mechanism generate query vectors that guide attention over the semantic space, playing a key role in semantic alignment. In contrast, W^V produces value vectors that encode token-level features and are aggregated via attention scores to produce the final output representations. Inspired by the aforementioned perspective, we conduct a comparative analysis of the W^Q and W^V projection matrices to investigate their parameter distribution patterns and variation trends in relation to downstream tasks. As illustrated in Figure 1, two notable phenomena can be observed: (1) the distribution of the W^Q matrices exhibits a high degree of dispersion after finetuning, and the Frobenius norms of ΔW^Q are relatively large, suggesting that W^Q across different layers captures diverse semantic information and is highly sensitive to downstream tasks; (2) in contrast, the W^V matrices display a highly concentrated and layer-consistent distribution, and the relatively small Frobenius norms of ΔW^V indicate more stable and task-invariant encoding behavior, reflecting their role in representing generalizable token-level features.

Inspired by the above analysis, we propose Function-aware **A**symmetric **I**nitialization for **Lo**w-**R**ank **A**daption (AILoRA), a parameter-efficient finetuning method that introduces an asymmetric initialization strategy to better align with the distinct functional roles of projection matrices in attention mechanism. Specifically, we perform singular value decomposition (SVD) on the pretrained W^Q and W^V matrices, and utilize the principal singular components (those with the largest singular values) of W^Q and the minor components (associated with the smallest singular values) of W^V to initialize their respective LoRA modules. This asymmetric initialization offers two key advantages: (1) it enables the LoRA modules of W^Q to rapidly adapt to the semantic space of downstream tasks and extract task-relevant semantic features, thereby facilitating more domain-sensitive attention computation; (2) it allows the LoRA modules of W^V to refine task-specific representations while preserving the generalizable feature encoding capabilities acquired during pretraining. We conduct comprehensive experiments across various model architectures, parameter scales, and datasets from diverse downstream tasks. The results demonstrate that AILoRA consistently outperforms mainstream PEFT methods in both performance and convergence speed. The main contributions are summarized as follows:

- To enhance the effectiveness of LoRA, we are the first to leverage the functional asym-

metry of attention projection matrices: W^Q captures task-sensitive semantic information essential for attention distribution, whereas W^V encodes more stable token-level features.

- Based on our empirical observations, we propose AILoRA, a novel PEFT method that introduces a function-aware asymmetric initialization strategy for LoRA modules, effectively striking a better balance between task-specific adaptability and the retention of pretrained knowledge.

- Comprehensive experiments across diverse model architectures, parameter scales, and downstream tasks demonstrate that AILoRA consistently surpasses existing PEFT baselines, while significantly accelerating convergence.

2 Related Works

2.1 Parameter-efficient Finetuning

Despite its success across numerous tasks, full finetuning still has several limitations. Notably, finetuning requires updating all parameters of pretrained models, which is impractical given the explosive growth of parameter amounts. Recent years have witnessed the rise of parameter-efficient finetuning methods, known as PEFT. PEFT techniques freeze most parameters and update only a small set of parameters to reduce computing resource consumption without compromising model performance. There are three mainstream classes of PEFT methods: addition-based, selection-based, and reparametrization-based (Lialin et al., 2023). The addition-based PEFT methods freeze pretrained weights and inject trainable parameters or modules, such as Adapter tuning (Houlsby et al., 2019), Prefix-tuning (Li and Liang, 2021) and Prompt tuning (Lester et al., 2021). The selection-based PEFT methods select a subset of parameters and freeze the rest, including BitFit (Zaken et al., 2022) and FAR (Vucetic et al., 2022). The reparametrization-based methods introduce reparametrization to reduce trainable parameters, such as LoRA and KronA (Edalati et al., 2022). Building upon these methods, numerous variants have subsequently emerged, including P-tuning (Liu et al., 2023), QLoRA (Dettmers et al., 2024) LoRA-drop (Zhou et al., 2025) and ComLoRA (Huang et al., 2025).

2.2 Low-Rank Adaptation

The low-rank adaptation (LoRA) is one of the most widely adopted PEFT techniques, grounded in the core assumption that the weight updates necessary for downstream task adaptation are intrinsically low-rank. Consequently, LoRA employs the product of two low-rank matrices, $A \in \mathbb{R}^{r\times n}$ and $B \in \mathbb{R}^{m\times r}$ to approximate the weight updates of $W \in \mathbb{R}^{m\times n}$. The model parameters can be expressed as

$$W = W_0 + \Delta W = W_0 + \frac{\alpha}{r} BA, \qquad (1)$$

where W_0 and ΔW denote the pretrained weights and weight updates, respectively. The scaling factor α is used to facilitate the optimization process, and r is the rank of two low-rank matrices ($r \ll min(m, n)$). The B matrix is initialized to all zero, while the A matrix adopts a random Gaussian distribution initialization. This initialization strategy ensures that $\Delta W = 0$ at the beginning of finetuning, implying no deviation from the pretrained weights. During finetuning, the pretrained weight W_0 keeps frozen and only the two low-rank matrices A and B are trainable.

3 Methodology

In this section, we present the details of the proposed method. Motivated by the observation that different projection matrices in the self-attention mechanism fulfill distinct semantic roles, we hypothesize that adopting a matrix-specific initialization strategy can better exploit their respective capacities. As highlighted in LASER (Sharma et al., 2023), the minor singular components of weight matrices contain noisy or long-tail information, while the principal singular components capture essential features across tasks. Consequently, we propose the function-aware **A**symmetric **I**nitialization for **Lo**w-**R**ank **A**daptation based on the unique properties of different singular components. The framework of AILoRA is illustrated in Figure 2. At first, AILoRA applies the singular value decomposition (SVD) to the pretrained weight matrices W^Q and $W^V \in \mathbb{R}^{m\times n}$. The SVD result is $W = U\Sigma V^T$, where $U = [u_1, u_2, \ldots, u_m] \in \mathbb{R}^{m\times m}$ and $V = [v_1, v_2, \ldots, v_n] \in \mathbb{R}^{n\times n}$ are the singular matrices with orthonormal columns and V^T is the transpose of V. $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix, where diagonal elements are the singular values arranged in descending order. Then, AILoRA uses the SVD results to initialize the LoRA modules.

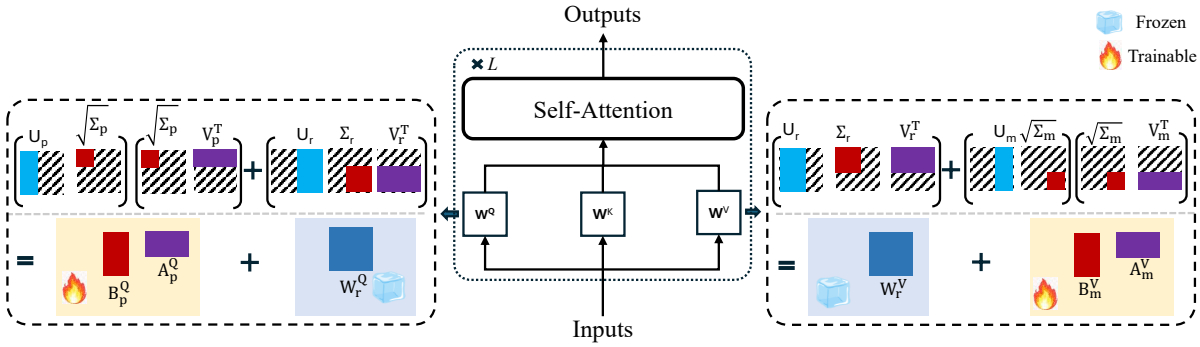


Figure 2: AILoRA first performs SVD on the W^Q and W^V matrices. For the W^Q matrices, the principal components are used to initialize the LoRA modules while keeping the remaining components frozen.In contrast, the LoRA modules of W^V are initialized using the minor components, with the remaining components fixed.

For the W^Q matrices, AILoRA utilizes the largest r singular values and their corresponding singular vectors to form the principal low-rank matrices A_p^Q and B_p^Q as illustrated in the left panel of Figure 2, which can be formulated as:

$$\begin{cases} B_p^Q = U_{[:,:r]}\Sigma_{[:r,:r]}^{1/2} \in \mathbb{R}^{m\times r}, \\ A_p^Q = \Sigma_{[:r,:r]}^{1/2} V_{[:,:r]}^T \in \mathbb{R}^{r\times n}. \end{cases} \tag{2}$$

And the remaining components are used to construct the residual matrices W_r^Q, which are frozen during finetuning:

$$W_r^Q = U_{[:,r:]}\Sigma_{[r:,r:]}V_{[:,r:]}^T \in \mathbb{R}^{m\times n}. \tag{3}$$

The matrix slicing notations used above are consistent with those in Python, in which [:r] denotes the first r dimensions. The low-rank matrices A_p^Q and B_p^Q can be multiplied to obtain the full-size principal matrices W_p^Q and the W^Q matrices can be formed:

$$W^Q = W_p^Q + W_r^Q = B_p^Q A_p^Q + W_r^Q. \tag{4}$$

The LoRA modules of the W^Q matrices contain knowledge that significantly influences attention computation. By training the principal components, the W^Q matrices can swiftly adapt to the semantic space of downstream tasks and perform a more domain-oriented computation of attention distribution.

For the W^V matrices, AILoRA utilizes the smallest r singular values and their corresponding singular vectors to construct the minor low-rank matrices A_m^V and B_m^V as presented in the right panel of Figure 2. The remaining components construct the residual matrices W_r^V, which is kept frozen during finetuning:

$$\begin{cases} B_m^V = U_{[:,-r:]}\Sigma_{[-r:,-r:]}^{1/2} \in \mathbb{R}^{m\times r}, \\ A_m^V = \Sigma_{[-r:,-r:]}^{1/2} V_{[:,-r:]}^T \in \mathbb{R}^{r\times n}, \\ W_r^V = U_{[:,:-r]}\Sigma_{[:-r,:-r]}V_{[:,:-r]}^T \in \mathbb{R}^{m\times n}. \end{cases} \tag{5}$$

The matrix slicing notations [-r:] denote the last r dimensions. Similarly, the low-rank matrices A_m^V and B_m^V are used to reconstruct the full-size minor matrices W_m^V and the W^V matrices can be formed:

$$W^V = W_m^V + W_r^V = B_m^V A_m^V + W_r^V. \tag{6}$$

The minor components of the W^V matrices, encapsulating less critical knowledge, are assigned to the low-rank matrices A_m and B_m. The optimization process enables the LoRA modules to master feature representations tailored to downstream tasks and mitigate the impact of noise. And the remaining components in the W_r^V matrices remain unchanged to preserve knowledge acquired by pretraining.

Inspired by LoRA+ (Hayou et al., 2024), we assign the LoRA module of the W^V matrices a learning rate twice that of the W^Q matrices, as the smaller magnitudes warrants a higher learning rate to ensure efficient convergence. The design of AILoRA similarly ensures no deviation from the pretrained weights at the beginning of training.

4 Experiments

To assess the effectiveness of AILoRA, we conduct extensive experiments on both Natural Language Understanding (NLU) and Natural Language Generation (NLG) tasks. The baselines include 1) LoRA, 2) PiSSA and 3) MiLoRA. All experiments are performed on a single NVIDIA A100 GPU unless otherwise specified.

Table 1: Model performance on the GLUE benchmark. The best results are shown in **bold**.

Model	Method	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B
RoBERTa-large (335M)	LoRA	67.9	90.2	92.7	94.3	**88.9**	85.8	96.2	91.8
	PiSSA	67.8	**90.3**	92.1	94.8	**88.9**	85.0	96.1	91.7
	MiLoRA	68.4	90.2	93.0	94.8	88.8	85.4	96.2	91.7
	AILoRA	**69.3**	**90.3**	**93.5**	**94.9**	88.8	**86.4**	**96.4**	**92.0**
DeBERTa-v3-base (184M)	LoRA	69.5	89.8	92.8	94.2	89.0	85.6	96.0	90.9
	PiSSA	68.8	89.0	92.2	94.1	88.5	84.3	96.0	90.9
	MiLoRA	68.8	89.5	92.8	94.2	88.9	**85.8**	95.8	**91.2**
	AILoRA	**70.5**	**90.0**	**93.3**	**94.3**	**89.1**	**85.8**	**96.1**	**91.2**

Table 2: Model performance with RoBERTa-large on SQuAD datasets. The best results are shown in **bold**

Dataset	Method	EM	F1
SQuAD v1.1	LoRA	**88.6**	**94.4**
	PiSSA	88.4	94.3
	MiLoRA	88.5	**94.4**
	AILoRA	88.4	94.3
SQuAD v2.0	LoRA	78.0	81.2
	PiSSA	77.4	81.1
	MiLoRA	77.5	81.2
	AILoRA	**78.5**	**82.2**

4.1 Experiments on NLU Tasks

Models and Datasets We finetune RoBERTa-large (Liu et al., 2019), an encoder-only model consisting of 24 layers, on the GLUE benchmark (Wang et al., 2018) and SQuAD datasets (Rajpurkar et al., 2016). The GLUE benchmark comprises nine natural language understanding tasks, covering single-sentence classification, similarity and paraphrase, and inference tasks. Consistent with prior researches, we exclude the WNLI task. Evaluation metrics also follow prior works: CoLA is evaluated using Matthew's Correlation, STS-B with Spearman's correlation coefficient, MRPC and QQP with F1 score, and the remaining tasks are evaluated using accuracy. The SQuAD datasets include two versions, SQuAD v1.1 and SQuAD v2.0, for which we report the Exact Match (EM) ratio and F1 score. Additionally, the encoder-only DeBERTa-v3-base with 12 layers (He et al., 2021) is also used on the GLUE benchmark for further comparison.

Implementations Details We use the implementation of transformers[1] for all NLU tasks. For the GLUE benchmark, the rank of low-rank matrices is uniformly set to 8 across all methods and datasets We conduct a grid search over learning rates and report the best results. Batch size, epoch number and other hyperparameters are consistent with PiSSA. For the SQuAD datasets, the numbers of epochs for SQuAD v1.1 and v2.0 are 3 and 2, respectively, with learning rates searched over {1e-4, 2e-4, 3e-4, 4e-4}. For fairness, all methods are evaluated using a uniform rank of 8. All experiments are repeated 5 times using random seeds, and the reported results are averaged over these runs.

Results We present the results on the GLUE benchmark and SQuAD datasets in Table 1 and Table 2, respectively. As shown in Table 1, our proposed AILoRA achieves the best performance on 15 out of 16 tasks. Notably, AILoRA exceed the best baseline by 0.6 and 1.0 points on the challenging task RTE and CoLA. In Table 2, AILoRA demonstrates competitive performance compared to baseline methods on SQuAD v1.1. On more challenging SQuAD v2.0, AILoRA outperforms all baselines, yielding improvements of 0.5 and 1.0 points in EM and F1 scores, respectively. In conclusion, these results highlight that AILoRA demonstrates enhanced adaptability to downstream tasks, which contributes to its superior performance across a variety of NLU tasks.

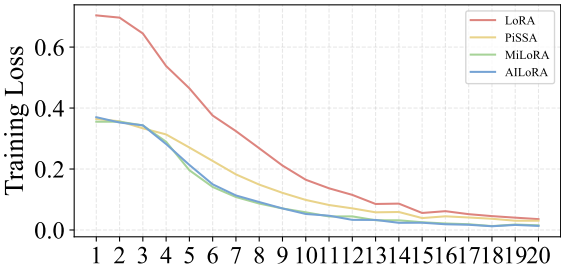
4.2 Experiments on NLG Tasks

Models and Datasets To evaluate the effectiveness of AILoRA on the NLG task, we finetune LLaMA2-7B (Touvron et al., 2023), a decoder-only model with 32 layers, on Math10K (Hu et al., 2023) dataset and evaluate its performance on five datasets, including 1) MultiArith (Roy and Roth, 2015), 2) GSM8K (Cobbe et al., 2021), 3) AddSub (Hosseini et al., 2014), 4) SingleEq

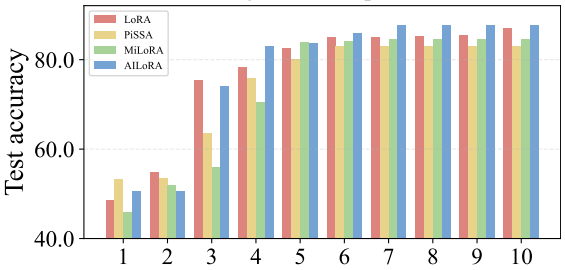
[1]https://github.com/huggingface/transformers

Table 3: Test accuracy with LLaMA2-7B on arithmetic reasoning tasks. The best results are shown in **bold**.

Method	MultiArith	GSM8K	AddSub	SingleEq	SVAMP	Avg.
LoRA	95.0	**41.8**	81.5	**86.6**	45.0	70.0
PiSSA	80.8	24.4	55.4	64.0	35.7	52.1
MiLoRA	**96.2**	40.7	80.0	84.4	**46.5**	69.6
AILoRA	95.8	40.5	**85.6**	86.4	**46.5**	**71.0**



(a) Training loss over epochs



(b) Test accuracy over epochs

Figure 3: The training loss and accuracy over the epochs of AILoRA and baselines.

(Koncel-Kedziorski et al., 2015) and 5) SVAMP (Patel et al., 2021). We report test accuracy for every dataset. Additionally, we fine-tune BART-large (Lewis et al., 2020) on the summarization task, and the details are provided in Appendix A.3.

Implementation Details For arithmetic reasoning, we use the implementation of LLM-Adapters (Hu et al., 2023). The AdamW optimizer (Loshchilov and Hutter, 2017) is employed with a learning rate 3e-4, warming up for 3% steps. We finetune LLaMA2-7B for 3 epochs and set the rank of low-rank matrices to 32 to accommodate the larger training corpus. All experiments are repeated 5 times on a single NVIDIA A800 GPU using random seeds to report the average results.

Results Table 3 presents the results on arithmetic reasoning. As illustrated, AILoRA achieves state-of-the-art performance across all datasets, surpassing the best baseline by 1.0 point on average. Notably, AILoRA outperforms the best baseline by 4.1 points on the AddSub dataset. The results demonstrate the scalability of AILoRA for NLG tasks.

Table 4: Comparison with more PEFT methods. The number of trainable parameters is reported to two decimal places. The best results are shown in **bold**.

Method	Params	CoLA	MRPC	RTE	STS-B
Full FT	355.36M	68.5	93.1	85.8	**92.1**
Adapter	7.40M	68.5	93.1	**87.0**	**92.1**
BitFit	1.32M	68.4	92.7	86.4	91.7
DoRA	0.84M	67.3	93.2	83.6	91.8
rsLoRA	**0.79M**	67.1	93.0	86.8	91.8
VeRA	1.11M	68.6	93.3	83.6	91.1
AILoRA	**0.79M**	**69.3**	**93.5**	86.4	92.0

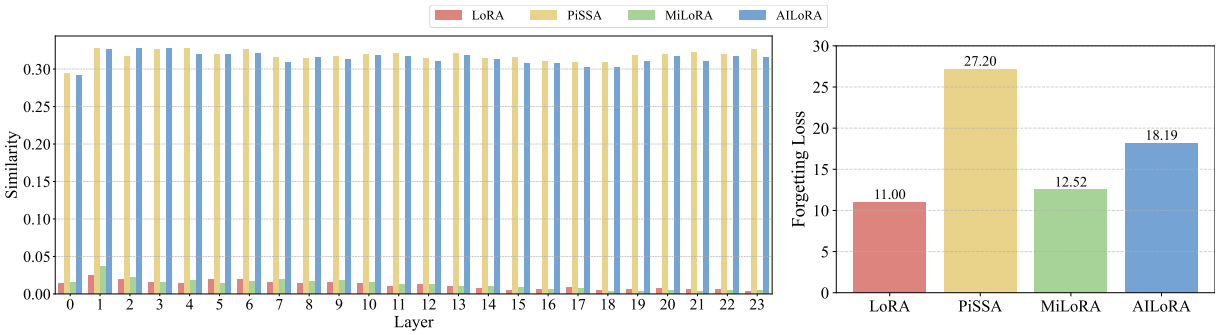
5 Ablation experiment

5.1 Convergence analysis

To assess the convergence behavior of AILoRA and baselines, we finetune RoBERTa-large on RTE dataset for 20 epochs with the rank r set to 8. Both training loss and test accuracy at each epoch are visualized as shown in Figure 3. As shown in Figure 3a, AILoRA consistently maintains the lowest training loss across epochs, indicating more efficient and stable optimization. Notably, LoRA exhibits a significantly higher loss after the first epoch, which highlights that the standard initialization strategy used in LoRA prevents efficient convergence at early stages. As illustrated in Figure 3b, AILoRA reaches 80% test accuracy within the first four epochs, faster than all baselines, and ultimately achieves the highest accuracy of 86.4%. These results further validate the effectiveness of AILoRA in accelerating convergence and enhancing overall performance.

5.2 Comparison with More PEFTs

To further assess the effectiveness of AILoRA, we compare AILoRA with full finetuning, classic and novel PEFT methods, including Adapter tuning, BitFit, DoRA (Liu et al., 2024), rsLoRA (Kalajdzievski, 2023) and VeRA (Kopiczko et al., 2023). Specifically, we finetune RoBERTa-large on four



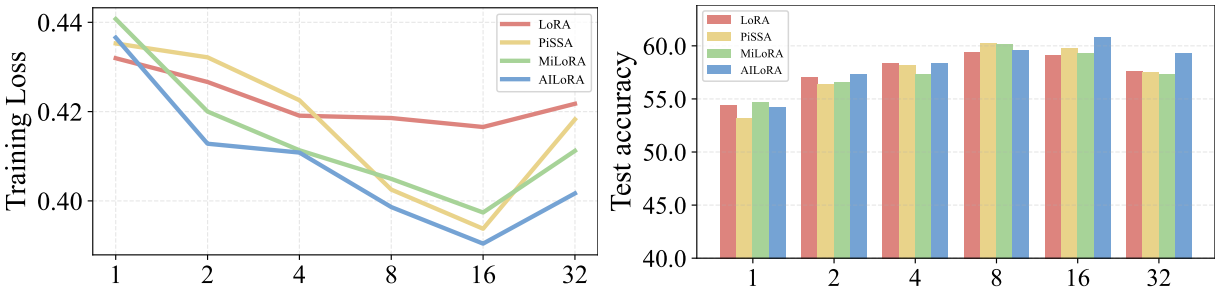
(a) Similarity Between low-rank matrices of W^Q and ΔW^Q across layers (b) Forgetting loss of AILoRA and baselines

Figure 4: Experiments on function-aware enhancement of W^Q and W^V.

tasks of GLUE benchmark. As for full finetuning, we adapt the hyperparameter configuration given in the original paper. Adapter tuning inserts and updates adapter modules into the self-attention and feedforward layers and the bottleneck size of adapters is set to 64 by default. BitFit only updates the bias terms of the pretrained model. DoRA decomposes pretrained weights into two components, direction and magnitude, and only applies LoRA to the direction component to enhance training stability. rsLoRA divides the LoRA modules by the square root of the rank, facilitating a straightforward finetuning compute/performance trade-off. VeRA shares a pair of frozen random matrices across all layers and conducts layer-wise adaptation using "scaling vectors". The rank of low-rank matrices of VeRA is set to 256, following the configuration used in the original paper. We repeat all experiments 5 times and report the best average results. As shown in Table 4, AILoRA achieves the highest scores on the CoLA and MRPC tasks by updating the fewest parameters, surpassing the best baselines by 0.7 point on CoLA. On RTE and STS-B tasks, AILoRA still achieves competitive results. Although Adapter tuning gets the highest scores, it updates $\times$6.7-9.3 more parameters. Overall, AILoRA achieves an optimal balance between parameter efficiency and model performance.

5.3 Experiments on Function-Aware Enhancement

To gain deeper insights into AILoRA, we conduct further experiments to investigate the function-aware enhancement of W^Q and W^V matrices. For W^Q matrices, we assess the similarity between the LoRA modules and ΔW after full finetuning, defined as the difference between the fully-finetuned and pretrained weights. The analysis follows the method outlined in LoRA. Specifically, we employ SVD to extract the first r columns of the left singular-vector matrices from both full-size low-rank matrices and ΔW. Then we compute the subspace similarity using the following metric: $\phi(A, B) = \frac{\|A^T B\|_F^2}{r}$, where $\|\cdot\|_F$ denotes the Frobenius norm. The value of $\phi(A, B)$ ranges from 0 to 1, with larger values indicating higher subspace similarity. As presented in Figure 4a, the LoRA modules of PiSSA and AILoRA exhibit strong similarity with ΔW, indicating that the knowledge they acquire closely resembles that obtained through full finetuning. This effect reveals that the LoRA modules of W^Q in PiSSA and AILoRA demonstrates strong adaptability to downstream task-specific semantic space. For W^V matrices, the cross-entropy loss, the usual next token prediction loss used when training LLMs, is used as the metric for measuring forgetting (Kalajdzievski, 2024). We measure the divergence between the predicted distributions of the pretrained model and the finetuned model to quantify the extent to which pretrained knowledge is forgotten after finetuning. Specifically, we finetune RoBERTa-large on the CoLA dataset and assess forgetting on the SST-2 test set. As shown in Figure4b, LoRA exhibits the lowest forgetting loss, as it does not modify the pretrained parameters, thereby minimizing the degree of forgetting of pretrained knowledge. By leveraging the minor components, MiLoRA introduces relatively minor perturbations to the pretrained parameters, and consequently incurs a relatively low forgetting loss. As for AILoRA and PiSSA, the primary distinction lies in that AILoRA utilizes minor components instead of principal ones (as in PiSSA) to initialize the LoRA of W^V. Consequently, AILoRA exhibits a significantly lower forgetting loss, demonstrating that AILoRA effectively preserves pretrained



(a) Training loss under various ranks

(b) Test MCC under various ranks

Figure 5: Comparison between AILoRA and baselines across various ranks.

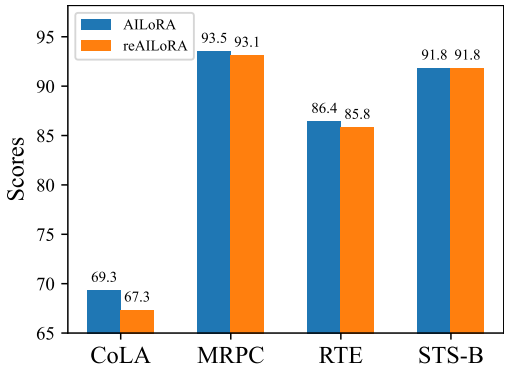


Figure 6: Model performance on the GLUE benchmark comparing AILoRA with reAILoRA.

knowledge.

5.4 Experiments on Initialization Strategies

To evaluate the validity of AILoRA's initialization scheme, we construct a control variant termed reAILoRA, where the LoRA modules are initialized in the reverse manner: using the minor components of W^Q and the principal components of W^V. We conduct experiments on CoLA, MRPC, RTE and STS-B tasks. As shown in Figure 6, AILoRA consistently outperforms reAILoRA across all tasks, validating the effectiveness of our function-aware asymmetric initialization. The principal components of W^Q primarily capture semantic-distinguishing knowledge, enabling the model to assign higher attention scores to contextually relevant tokens. Meanwhile, the minor components of W^V assign less-optimized knowledge to master feature representations of downstream tasks and mitigate the impact of noise.

5.5 Experiments on Various Ranks

In this section, we investigate the impact of increasing the rank from 1 to 32, aiming to assess whether AILoRA consistently outperforms baselines across different rank settings. The experiments are conducted on the CoLA dataset for 20 epochs, and the training loss of the training set and the accuracy on the test set are depicted in Figure 5. In Figure 5a, the training losses of AILoRA are almost the lowest compared to all baselines, indicating the best adaptability to downstream tasks. Figure 5b further demonstrates that AILoRA consistently surpasses all baselines under the same parameter budget, highlighting its broad adaptability and scalability. Notably, when the rank is increased to 32, both the training loss and test Matthews Correlation Coefficient (MCC) exhibit anomalous behavior: the training loss rises and the MCC declines. This observation highlights that simply increasing the number of trainable parameters does not guarantee improved performance and may even lead to degradation.

6 Conclusion

In this paper, we systematically investigate the distinct functional roles of the W^Q and W^V projection matrices in the attention mechanism. Motivated by these insights, we propose **AILoRA**, a novel parameter-efficient fine-tuning method. AILoRA introduces a function-aware asymmetric initialization scheme, leveraging the principal components of W^Q and the minor components of W^V to initialize the respective LoRA modules. This design enables LoRA to better exploit the functional asymmetry of attention matrices, leading to improved downstream performance and faster convergence. Extensive experiments across diverse model architectures (encoder-only, decoder-only, and encoder-decoder), parameter scales (ranging from 184M to 7B), and downstream tasks (including both NLU and NLG benchmarks) consistently validate the effectiveness and robustness of AILoRA.

Limitations

Due to hardware resource limitations, we are unable to conduct experiments on larger models, such as LLaMA2-13B. While AILoRA improves model performance and accelerates convergence, it does not reduce the number of trainable parameters. Future work will focus on reducing the number of trainable parameters while improving model performance.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

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

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. 2022. Krona: Parameter efficient tuning with kronecker adapter. *arXiv preprint arXiv:2212.10650*.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. In *International Conference on Machine Learning*, pages 17783–17806. PMLR.

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

Xinyi He, Yihao Liu, Mengyu Zhou, Yeye He, Haoyu Dong, Shi Han, Zejian Yuan, and Dongmei Zhang. 2025. Tablelora: Low-rank adaptation on table structure understanding for large language models. *arXiv preprint arXiv:2503.04396*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 523–533.

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

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

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. 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*, pages 5254–5276.

Qiushi Huang, Tom Ko, Lilian Tang, and Yu Zhang. 2025. Comlora: A competitive learning approach for enhancing lora. In *The Thirteenth International Conference on Learning Representations*.

Hamish Ivison, W Matthew, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, and 1 others. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.

Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*.

Damjan Kalajdzievski. 2024. Scaling laws for forgetting when fine-tuning large language models. *arXiv preprint arXiv:2401.05605*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Dawid J Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2023. Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart:

Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

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 *Forty-first International Conference on Machine Learning*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. Gpt understands, too. *AI Open*.

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

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. 2024. 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)*, pages 11662–11675.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary. *Topic-Aware Convolutional Neural Networks for Extreme Summarization. ArXiv abs/1808.08745*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1743–1752.

Pratyusha Sharma, Jordan T Ash, and Dipendra Misra. 2023. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. *arXiv preprint arXiv:2312.13558*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Danilo Vucetic, Mohammadreza Tayaranian, Maryam Ziaeefard, James J Clark, Brett H Meyer, and Warren J Gross. 2022. Efficient fine-tuning of bert models on the edge. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1838–1842. IEEE.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel 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.

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. 2024a. Milora: Harnessing minor singular components for parameter-efficient llm finetuning. *arXiv preprint arXiv:2406.09044*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *11th International Conference on Learning Representations, ICLR 2023*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Hongyun Zhou, Xiangyu Lu, Wang Xu, Conghui Zhu, Tiejun Zhao, and Muyun Yang. 2025. Lora-drop: Efficient lora parameter pruning based on output evaluation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5530–5543.

A Appendix

A.1 Experiments on Weight Matrix Selection

Given a limited parameter budget, which types of rank settings yield the best performance? To answer the above question, we conduct experiments on four tasks of GLUE benchmark, limiting the total number of trainable parameters to 0.79M on RoBERTa-large. For simplicity and parameter-efficiency considerations, we only apply low-rank adaptation to the attention weights and keep the FFN modules frozen. This parameter budget is equivalent to $r = 8$ if the low-rank adaptation is applied to two types of weight matrices and $r = 16$ when applied to one type. When adapting to the W^Q and W^V weights, we employed the asymmetric initialization method of AILoRA. For other weight types, the default initialization method of LoRA is used. The experimental results are summarized in Table 5. Notably, allocating the entire parameter budget to the W^K weights almost results in the poorest performance, as previously highlighted in LoRA. Moreover, applying low-rank adaptation to more than one type of weight matrices generally leads to better performance. At last, applying AILoRA to the W^Q and W^V weights yields the best results, demonstrating the effectiveness of our weight matrices selection.

Table 5: Results on different rank settings. The best results are shown in **bold**.

Rank Settings	CoLA	MRPC	RTE	STS-B
r_q=r_v=8 (AILoRA)	**69.3**	**93.5**	**86.4**	91.8
r_q=16	62.1	92.2	82.0	90.5
r_k=16	58.7	90.5	79.2	89.7
r_v=16	66.5	92.1	85.9	91.9
r_o=16	65.4	91.7	84.8	**92.0**
r_q=r_k=8	62.6	90.8	83.1	90.4
r_k=r_v=8	67.7	89.5	85.8	91.7
r_q=r_k=r_v=r_o=4	68.5	89.0	85.6	91.9

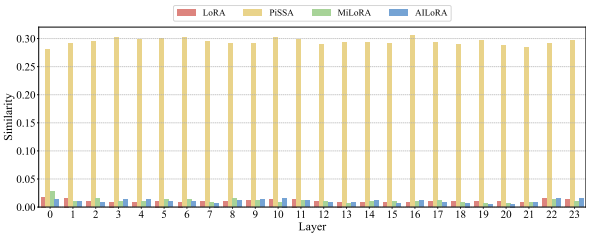


Figure 7: The similarity between LoRA modules with ΔW of Value.

A.2 The subspace similarity of W^V matrices

To further analyze the functional enhancement of AILoRA, we compare the subspace similarity between the LoRA module weights and ΔW for the W^Q matrices. As presented in Figure 7, PiSSA exhibits stronger subspace alignment with ΔW, owing to direct optimization over the principal singular components of pretrained weights. In contrast, the random and all-zero initialization of LoRA results in relatively small updates, leading to lower similarity with ΔW. MiLoRA and AILoRA specifically optimize the minor components of the pretrained weights, enabling the model to acquire task-relevant feature representations during finetuning while preserving the knowledge obtained through pretraining. Therefore, the subspace similarities of MiLoRA and AILoRA are likewise notably low.

A.3 Experiments on Summarization Tasks

We finetue BART-large, which adopts an encoder&decoder architecture with 12 encoder layers and 12 decoder layers, on two summarization datasets: XSum (Narayan et al., 2018) and CNN/DailyMail (Hermann et al., 2015), and evaluate model performance using Rouge 1/2/L scores (R-1/2/L, (Lin, 2004)). We use the implementation of transformers and follow the setting of AdaLoRA

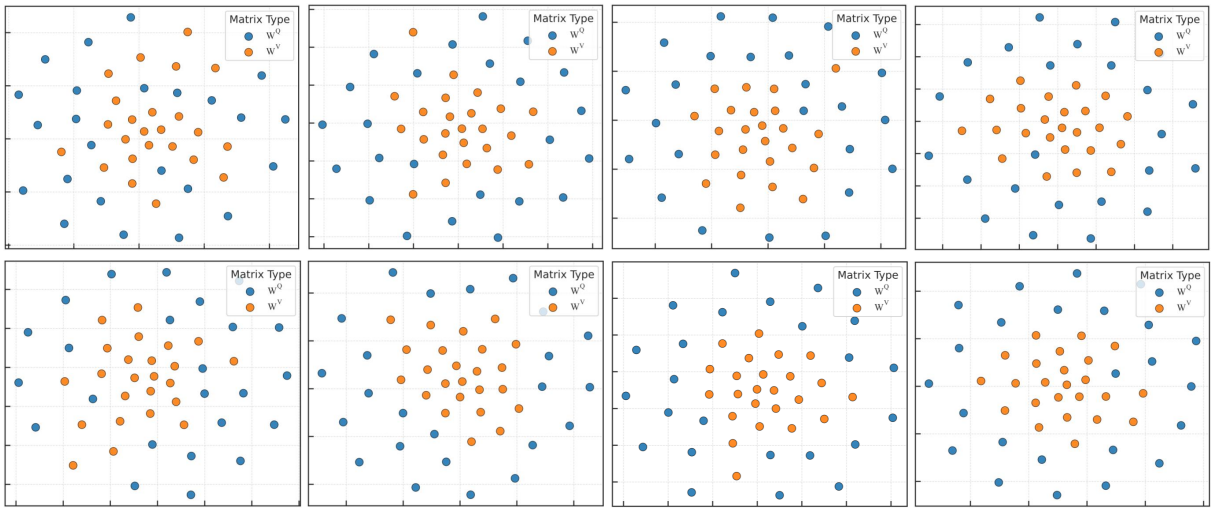
Table 6: Results with BART-large on summarization tasks. We report R-1/2/L scores. The best results are shown in **bold**.

Method	XSUM	CNN/DailyMail
LoRA	40.46 / 17.55 / 32.36	42.73 / 19.75 / 29.17
PiSSA	40.63 / **17.63** / **32.51**	42.74 / 19.65 / 29.18
MiLoRA	40.31 / 17.35 / 32.15	42.84 / 19.78 / 29.17
AILoRA	**40.66** / 17.61 / **32.51**	**42.91** / **19.79** / **29.24**

(Zhang et al., 2023), setting the rank of low-rank matrices to 8 and training epochs to 15. And we set the beam length as 8 and batch size as 64, while for CNN/DailyMail, we set the beam length as 4 and batch size 32. We conduct a grid search over learning rates in {5e-5, 1e-4, 5e-4} and report the best results. Results shown in Table 6 demonstrate that AILoRA consistently achieves best scores across all matrics, closely matching or surpassing the best-performing baselines.

A.4 The visualization of AILoRA and baselines

To further investigate the impact of the initialization methods on LoRA modules, we employ the t-SNE technique on AILoRA and baselines to Visualize the finetuned weights. As depicted in Figure 8, it can be seen that the distribution of LoRA can not be easily distinguished. The low-rank matrix B is set to all zero and A is randomly initialized, resulting in small updates and negligible impact on the discrimination. In contrast, the distribution of PiSSA, MiLoRA and AILoRA are distinctly separated. The low-rank matrices of there method are initialized using singular components of pretrained weights, contributing to larger update magnitudes and thus more easily distinguishable distribution. Although The distribution of AILoRA resembles that of PiSSA and MiLoRA, AILoRA uses the singular components of W^Q to rapidly adapt to the semantic space of downstream tasks, and utilizes the minor components of the W^V matrices to grasp downstream-task-specific feature representations during finetuning and the pretrained knowledge is well preserved. Consequently, AILoRA generally outperforms PiSSA and MiLoRA.



(a) LoRA (b) PiSSA (c) MiLoRA (d) AILoRA

Figure 8: Visualization results of the low-rank matrices on CoLA dataset.