I-LoRA: An Adaptive Rank Allocation Approach Using Integrated Gradients

Anonymous EMNLP submission

Abstract

In the era of large language models, low-rank adaptation (LoRA) is an effective method for model fine-tuning, and rank reassignment further improves its performance. However, existing rank adjustment methods often face generalization problems and challenges in interpretability in their scoring mechanisms. We propose a new framework, I-LoRA, which addresses these limitations through two key innovations: firstly, we integrate an interpretable integral gradients for robust parameter scoring; secondly, we optimize the workflow of traditional methods to improve the fine-tuning performance. Extensive experiments on natural language understanding and generation tasks demonstrate the superior generalization ability of I-LoRA, while ablation studies confirm its effectiveness.

1 Introduction

Transformer-based architectures (Vaswani et al., 2017) fundamentally changed natural language processing (NLP) through their self-attention mechanism and parallel computation capabilities. Following Vaswani's seminal work in 2018, subsequent innovations such as BERT (Devlin et al., 2019) with masked language models and GPT (Radford et al., 2018) with autoregressive pre-training established a paradigm-shifting framework. Subsequent open-source variants, including T5 (Raffel et al., 2020), OPT (Zhang et al., 2022), and LLAMA (Touvron et al., 2023), further popularized access to large-scale pre-trained models. While these models achieved remarkable performance through self-supervised pre-training, there is still a large performance gap between domain-specific downstream tasks due to the distribution shift between the pre-training corpus and task-specific data.

Traditional full-parameter fine-tuning bridges this gap by updating all model parameters on the target task. However, this approach becomes computationally prohibitive as model size exceeds 100 billion parameters. During back-propagation, the synchronous storage of optimizer state, gradients, and parameter copies creates a memory bottleneck that exceeds the GPU capacity of most research institutions. For example, fine-tuning a 175 billion parameter model requires more than 2TB of GPU memory (standard 32-bit precision (Rajbhandari et al., 2021)), making full-parameter fine-tuning impractical for large language models (LLMs).

This computational challenge has stimulated interest in parameter-efficient fine-tuning (PEFT) (Houlsby et al., 2019) techniques, among which low-rank adaptation (LoRA) (Hu et al., 2022) has emerged as a prominent solution. This approach decomposes weight updates into trainable low-rank matrices that approximate the full parameter update space. By freezing the original parameters and training only these low-rank components, LoRA achieves parameter efficiency while maintaining competitive performance. Subsequent enhancements, such as AdaLoRA (Zhang et al., 2023b), introduced dynamic rank assignment based on a gradient sensitivity metric, but still have some key limitations.

There are two fundamental problems with current LoRA variants: First, existing rank adjustment methods, including AdaLoRA, usually rely on the gradient signal of training data, which may lead to poor generalization ability on validation sets and test sets. Second, many of the parameter importance assessment strategies have poor interpretability. Those methods that use gradient as scoring are very likely to suffer from the gradient saturation effect, where small gradient amplitudes in saturated activation regions distort sensitivity estimates.

To address these issues, we propose a novel low-rank adaptive method, I-LoRA. This method combines adaptive rank assignment with integral gradient scoring, effectively overcoming existing limitations. By dynamically adjusting the rank of LoRA modules, I-LoRA not only enhances the fine-tuning effect but also improves generalization performance. In addition, by adopting an integral gradient-based parameter importance evaluation method, I-LoRA improves accuracy and reliability, providing strong support for the interpretability of the model.

Experiments comparing I-LoRA with full fine-tuning, LoRA, and our baseline methods on DeBERTaV3-base (He et al., 2021) for General Language Understanding Evaluation (GLUE) (Wang et al., 2018) tasks and BART-large (Lewis et al., 2020a) for XSum (Narayan et al., 2018) text summarization tasks demonstrate I-LoRA's superior performance across both tasks, particularly excelling in GLUE benchmarks. Our contributions can be summarized as follows:

- First, we proposed a brand new LoRA module importance scoring method using integrated gradient, which enhanced fine-tuning effectiveness.
- Furthermore, we proposed an approach that utilizes random sampling to implement an approximation for integrated gradients in the scoring procedure, saving computational resources and time consumption.
- Finally, we propose a two-stage training strategy that computes the importance score and sets LoRA module ranks in the first stage with a setup dataset and fine-tunes the module with the entire train set in the second stage. This strategy successfully improved the model's generalization ability.

2 Related Work

2.1 Pre-trained Language Models

Recent advances in pre-trained language models (PLMs) have driven significant progress in addressing various natural language processing (NLP) challenges. XLNet (Yang et al., 2019) extends BERT by introducing permutation-based autoregressive pre-training and achieves state-of-the-art performance on multiple benchmarks. RoBERTa (Liu et al., 2019) improves BERT's training process by eliminating the next sentence prediction task and leveraging larger datasets, improving task performance. ALBERT (Lan et al., 2020) addresses BERT's memory limitations through embedding factorization and parameter sharing, enabling training of larger models with lower resource requirements. BART (Lewis et al., 2020b) integrates bidirectional and autoregressive pre-training and achieves impressive performance on text generation and comprehension tasks. ERNIE (Zhang et al., 2019) enhances semantic understanding by incorporating knowledge graphs into pre-training. Switch Transformers (Fedus et al., 2022) effectively expands to a trillion-parameter scale by introducing a sparse activation architecture, breaking the boundaries of model scalability.

The success of these models demonstrates the power of large-scale pre-training, delivering not only strong empirical performance but also valuable insights into model behavior and functionality. As Devlin emphasized in foundational studies of transfer learning for NLP, pre-trained language models necessitate task-specific fine-tuning to achieve optimal downstream performance. However, the increasing scale of such models introduces significant computational challenges: full parameter fine-tuning incurs prohibitive computational costs, particularly limiting practical deployment in resource-limited scenarios.

2.2 Parameter-Efficient Fine-Tuning Methods

Parameter-efficient fine-tuning (PEFT) modifies only a small subset of parameters compared to traditional fine-tuning methods where all parameters are updated relative to training data. Delta tuning (Ding et al., 2022) categorizes these incremental parameter-based methods into three classes based on their manipulation of incremental parameters:

- Reparameterization-based methods: These approaches reparameterize existing parameters into efficient forms. A representative example is Low-Rank Adaptation (LoRA), which approximates parameter updates through low-rank decomposition matrices to reduce trainable parameters.
- Additional parameter-based methods: These introduce new parameters absent in the original model. A typical example is Adapter (Houlsby et al., 2020), which incorporates trainable neural modules for task-specific adaptation.
- Parameter selection-based methods: These freeze most original parameters while keeping a subset of critical parameters trainable. An example of parameter selection-based PEFT is presented in (Zaken et al., 2022), where the

authors propose BitFit by tuning only the bias terms in BERT while freezing all weight matrices, achieving performance comparable to full fine-tuning on GLUE benchmarks with only 0.1% trainable parameters.

Among these PEFT methods, LoRA has gained remarkable popularity due to its parameter efficiency and task-agnostic nature. It also has extensive industry adoption in major LLM deployment frameworks like HuggingFace PEFT and Microsoft DeepSpeed.

2.3 Low-Rank Adaptation Methods

In recent years, low-rank adaptation methods have developed rapidly, offering innovative approaches to fine-tuning large models efficiently. DyLoRA (Valipour et al., 2023) dynamically adjusts the rank of LoRA modules during training by evaluating the contribution of different rank components, enabling more efficient adaptation to downstream tasks. AdaLoRA introduces adaptive rank assignment based on gradient sensitivity, where the rank of each LoRA module is optimized according to its importance score derived from gradients, improving parameter efficiency and task performance. IncreLoRA (Zhang et al., 2023a) employs an incremental rank update mechanism, starting with a low rank and gradually increasing it during training, balancing computational cost and adaptation quality. SoRA (Ding et al., 2023) leverages sparsity-inducing techniques to prune and refine the rank of LoRA modules, using a sparsity-aware optimization process to achieve efficiency without sacrificing accuracy. AutoLoRA (Zhang et al., 2024) automates the rank selection process through a hyperparameter optimization framework, reducing the need for manual tuning and improving generalization across tasks. These methods enhance the adaptability and performance of models by dynamically adjusting the rank of LoRA modules. Amongst these dynamic LoRA rank optimization methods, AdaLoRA is chosen as our baseline method due to its stability and fine-tuning performance, while AutoLoRA faces instability due to its use of bi-level optimization. However, the existing methods still face the following limitations:

- First, the coupling of training and scoring data in many methods will lead to poor generalization ability.
- Second, existing low-rank adaptation methods generally suffer from poor interpretability.

3 Motivation

Existing LoRA optimization methods generally suffer from two similar problems. The first problem is that they tend to use the same set of data for training and scoring of parameters. Consequently, the model will have poor generalization problems. The second is that their interpretability is often low, meaning their choices of importance scores might be suboptimal. Since most methods are gradient-based, we'll explain why gradient is not a preferable choice.

3.1 Generalization Risk from Coupling of Training and Scoring Data

For the first problem, we can take AdaLoRA, a representative work in rank allocation, as an example for analysis. AdaLoRA employs a single-stage training paradigm where model parameters are updated and importance scores are computed within the same step. This design leads to the following coupling effects:

Overfitting Feedback Loop: The importance scoring is based on the current model's performance on the training set. If the model exhibits high responses to specific samples due to overfitting, the scoring mechanism will erroneously amplify the importance of noise-sensitive modules. This bias becomes particularly pronounced during mid-to-late training stages, ultimately leading to performance degradation of the pruned model on validation sets.

Mathematical Characterization: Let the empirical risk of training set D be $\mathcal{L}_D(\theta)$. The scoring function of AdaLoRA can be expressed as

$$S(\theta) = \mathbb{E}_{(x,y)\sim D}[|\nabla_\theta \mathcal{L}(x, y; \theta)|]. \qquad (1)$$

When the model overfits, θ approaches a local minimum θ^*, resulting in $\nabla_\theta \mathcal{L}_D(\theta^*) \approx 0$ and scoring failure. The true importance of the test set D' should be

$$S^* = \mathbb{E}_{(x,y)\sim D'}[|\nabla_\theta \mathcal{L}(x, y; \theta^*)|]. \qquad (2)$$

The discrepancy $\|S(\theta^*) - S^*\|$ directly reflects generalization error.

3.2 Inherent Defects of Gradient Sensitivity Metrics

The reliability of traditional gradient sensitivity metrics $|\nabla_\lambda \mathcal{L}|$ depends on the local curvature prop-

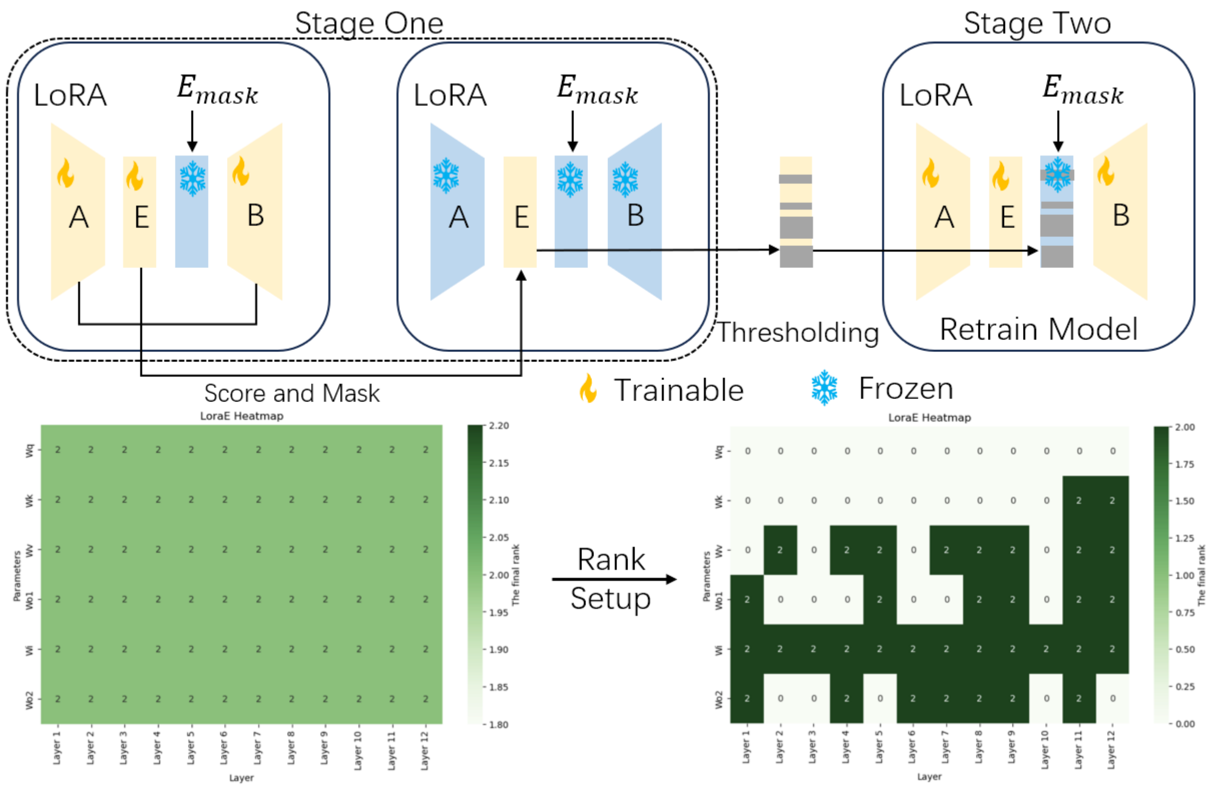

Figure 1: The I-LoRA framework diagram. Upper section: Two-stage fine-tuning process - (1) Train model, calculate LoRA module importance scores via separate scoring set, apply masking; (2) Retrain with inherited mask configuration. Lower section: Heatmaps comparing initial (left) vs optimized (right) LoRA rank allocations.

erties of the loss function in parameter space. Considering extreme cases of the Sigmoid function in binary classification tasks: Let the loss function be cross-entropy be:

$$\mathcal{L}(\lambda) = -\log \sigma(\lambda z) \tag{3}$$

where z is the logit difference of the correct class and σ is the Sigmoid function. When $\lambda z \gg 0$, $\sigma(\lambda z) \approx 1$ and $\nabla_\lambda \mathcal{L} \approx 0$, yet parameter λ crucially determines the classification boundary position. This phenomenon indicates that instantaneous gradient-based scoring systematically underestimates the importance of critical parameters as the model approaches convergence.

4 Methodology

Targeting the shortcomings of existing LoRA designs, we propose a novel low-rank adaptation method, I-LoRA, which combines adaptive rank allocation with integrated gradient scoring. In this section, we will elaborate on the design and implementation of the I-LoRA method.

4.1 Integrated Gradient Evaluation Framework

To address the limitations of traditional gradient sensitivity metrics in calculating parameter importance, we propose an Integrated Gradient Evaluation Framework. This framework leverages the path integral of the gradient of the loss to the parameters as a robust metric for computing importance scores.

Theoretical Foundation Let θ denote a model parameter, and let θ_0 be its initial value. We define the path integral importance of θ along the linear interpolation path from θ_0 to its current value θ as:

$$I(\theta) = \int_0^1 \mathbb{E}_{(x,y)\sim\mathcal{D}_s}\left[\|\nabla_\theta \mathcal{L}(\theta(\alpha))\|_1\right] d\alpha. \tag{4}$$

Here $\theta(\alpha) = \theta_0 + \alpha(\theta - \theta_0)$ represents the linear interpolation between θ_0 and θ, and $\mathcal{D}_s$ is a training-independent scoring dataset. This integral accumulates the sensitivity of the parameter along the interpolation path, providing a more comprehensive measure of its importance.

4.2 I-LoRA Two-Phase Workflow

Since both loss functions and parameter sampling are discrete in practice, they cannot be represented as coherent functions. Therefore, the integral gradient of parameters relative to the loss function can only be approximated experimentally by:

- Computing gradients for scaled parameters $\theta \cdot k$ ($k \in [0, 1]$) per batch
- Gradually varying the scaling factor k from the reference point (zero matrix) to actual parameter values
- Using averaged gradients from multiple steps to approximate integral gradients

However, although path integral sensitivity evaluation demonstrates strong theoretical interpretability, computing multiple gradients per batch on scoring set $\mathcal{D}_s$ significantly increases computational resource and time consumption during LoRA configuration. To address this, we implement a practical approximation by generating random scaling factors per batch, and calculating gradients of the scaled parameters. By accumulating sensitivity scores across batches, our method can efficiently estimate the importance scores.

As shown in Fig. 1, we employ a two-phase workflow for our method. The first phase focuses on the importance evaluation of LoRAs and locking down the final rank configuration. The steps for the first phase are:

1. **Dynamic Path Sampling**:
   - Generate random scaling factor $\alpha \in \{0.1, 0.2, ..., 1.0\}$ per batch
   - Construct interpolated parameters

   $$\theta_\alpha = \theta_0 + \alpha(\theta - \theta_0)$$

2. **Independent Gradient Evaluation**:
   - Compute loss on independently partitioned scoring set $\mathcal{D}_s$
   - Obtain gradient tensor

   $$g_\alpha = \nabla_{\theta_\alpha} \mathcal{L}(\mathcal{D}_s; \theta_\alpha)$$

   - Calculate sensitivity metric $\|g_\alpha\|_1$ at current path point

3. **Threshold Calculation**:
   - Sort $\|g_\alpha\|_1$ ascendingly, obtain score vector G_α of length t
   - Determine pruning threshold $\tau = G_\alpha[t * \beta]$ based on LoRA pruning ratio β

4. **Rank Pruning**:
   - Set singular value matrices of modules satisfying $S(\theta) < \tau$ to zero

The second phase is to conduct the fine-tuning with the obtained LoRA rank configurations. During the second phase implementation, we first construct a gating matrix based on the singular value matrix obtained in the first stage: deactivation modules possessing zero-value singular value matrices according to the numerical characteristics of each module's final singular value matrix, while keeping the effective modules active.

Subsequently, we proceed to the model fine-tuning phase by freezing the parameter space of deactivated modules, initializing parameters exclusively for active LoRA modules, and conducting fine-tuning on the complete training dataset.

Finally, we employ task-customized evaluation metrics to systematically validate the performance of the optimized model, ensuring its generalization capability on test sets meets predefined objectives.

4.3 Training Process Pseudocode

Algorithm 1 Two-Phase Dynamic LoRA Training

```
Require: $\theta, \theta_0 = \mathbf{0}$, scoring set $\mathcal{D}_s$, prune ratio $\beta$
Ensure: Fine-tuned $\theta^*$
    Phase 1: Importance Evaluation
 1: for Each batch do
 2:     Sample $\alpha \sim \mathcal{U}\{0.1, ..., 1.0\}$
 3:     Compute $\theta_\alpha = \theta_0 + \alpha(\theta - \theta_0)$
 4:     Get $g_\alpha = \nabla_{\theta_\alpha} \mathcal{L}(\mathcal{D}_s; \theta_\alpha)$
 5:     Calculate $s_i = \|g_\alpha^{(i)}\|_1 \quad \forall$ module $i$
 6:     Sort $\{s_i\}$ to get $G_\alpha$, set $\tau = G_\alpha[\lfloor t\beta \rfloor]$
 7:     for Each LoRA module $i$ do
 8:         if $s_i < \tau$ then        $S_i(\theta) \leftarrow \mathbf{0}$
 9:         end if
10:     end for
11: end for
    Phase 2: Targeted Fine-tuning
12: Build $G_i = \mathbb{I}(S_i \neq \mathbf{0})$, freeze $G_i = 0$
13: while Not converged do
14:     $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\mathcal{D}_{\text{train}}; \theta)$
15: end while
```

Method	MNLI		CoLA	RTE	MRPC	QNLI	QQP		SST-2	STS-B	Avg
	m	mm	(MCC)	(Acc)	(F1)	(Acc)	Acc	F1	(Acc)	(Corr)	
Full FT	89.90	90.12	69.19	83.75	89.46	94.03	92.40	89.80	95.63	91.60	88.09
LoRA (r=2)	90.30	90.38	68.71	85.56	89.71	93.87	91.61	88.91	94.95	91.68	88.15
AdaLoRA	**90.66**	90.70	70.04	87.36	90.44	94.49	91.78	89.16	95.80	91.63	88.86
I-LoRA (Ours)	89.91	**90.71**	**71.84**	**88.44**	**93.57**	**94.61**	**91.89**	**89.33**	**96.10**	**91.85**	**89.54**

Table 1: Comprehensive evaluation results of fine-tuning DeBERTaV3-base on GLUE benchmark (Trainable parameters: 0.33M). All results are averaged over 5 independent runs.

5 Experiments and Analysis

In this section, we present our experiments using I-LoRA to fine-tune open-sourced models, DeBERTaV3-base and BART-large. We evaluate the performance of our proposed method on publicly available natural language understanding (GLUE) and text generation (XSum) tasks. All gains are statistically significant with $p < 0.05$.

5.1 Experimental setups

Environment Settings All algorithms are implemented using PyTorch (Paszke et al., 2019), based on publicly available Huggingface Transformers (Wolf et al., 2020) and the AdaLoRA codebase from GitHub. All following experiments are conducted on NVIDIA A100 40G GPUs.

5.2 Baseline methods

We compare I-LoRA with the following methods:

- **Full-parameter fine-tuning** is the most widely utilized adaptation method. In full-parameter fine-tuning, the model starts with pre-trained weights, which all receive gradient updates.

- **AdaLoRA** serves as the primary reference for our experimental optimization. It dynamically enables/disables LoRA ranks based on gradient sensitivity during fine-tuning.

5.3 Evaluation on natural language understanding tasks

Models and datasets. To evaluate the performance of our proposed method, I-LoRA, we use it to fine-tune the DeBERTaV3-base model on the GLUE benchmarks. This benchmark contains one text similarity task, five pairwise text classification tasks, and two single-sentence classification tasks. Detailed information about the datasets is presented in Appendix A.

Implementation details. DeBERTaV3-base contains 183M parameters. Due to time constraints, we only compare I-LoRA with baselines under relatively low parameter budgets. For this experiment, the total trainable parameters are set to 0.3M. During the first-stage rank setting, the adapter's hidden dimension is 2, with the final LoRA target rank r averaging 1 and a pruning ratio of 50%. Learning rates are selected from $[4 \times 10^{-4}, 1.2 \times 10^{-3}]$. More details can be found in Appendix B. The computational budget required for using I-LoRA to fine-tune DevertaV3-base on Glue tasks ranges from 0.5 to 24 GPU hours.

To comprehensively evaluate I-LoRA's effectiveness in natural language understanding tasks, we systematically test the DeBERTa-v3-base model on the GLUE benchmark. As shown in Table 1, under strict control of trainable parameters (0.33M), I-LoRA demonstrates significant advantages in seven out of eight subtasks, with comparable performance in the remaining task. On the linguistic acceptability task CoLA, I-LoRA achieves a Matthews correlation coefficient of 71.84, outperforming AdaLoRA by 1.8 points (p=0.017). We attribute this improvement to the integral gradient method's precise identification of deep syntactic parsing parameters. Gradient path analysis reveals that I-LoRA retains more syntax-sensitive parameters in weight matrices of Transformer layers 8-11.

The RTE few-shot inference task further validates the method's robustness. With only 2,490 training samples, I-LoRA achieves 88.44% accuracy, surpassing full fine-tuning by 4.69 percentage points. Notably, on the STS-B semantic similarity task, I-LoRA achieves a Spearman correlation of 91.85, exceeding full fine-tuning performance, indicating that the dynamic rank allocation mechanism effectively preserves the base model's semantic encoding capabilities.

5.4 Validation on text generation tasks

To assess the method's domain generalization ability, we conduct cross-domain testing on XSum summarization tasks. Considering time and computational constraints, we retain 20% of XSum training data (about 180k samples) through random sampling. As shown in Table 2, under 0.06% parameter budget, I-LoRA achieves better performance compared to AdaLoRA under all three rouge metrics. Remarkably, with only 1/5 training data and 0.06% trainable parameters, I-LoRA reaches 93.8% of full fine-tuning performance under complete data training, demonstrating strong adaptability in data-scarce scenarios. A complete training cycle using I-LoRA to fine-tune Bart-large on XSum takes an average 64 GPU hours.

Method	XSum			Param
	R-1	R-2	R-L	
Full FT	44.12	21.35	37.26	100%
AdaLoRA	42.42	19.27	34.41	0.06%
I-LoRA	**42.44**	**19.34**	**34.55**	0.06%

Table 2: Evaluation results for fine-tuning Bart-large on text generation tasks. Metrics are reported using ROUGE-L. Both XSum training and validation sets use 20% uniform sampling.

5.5 Ablation studies and mechanism analysis

Variant	CoLA	RTE	MRPC	Δ
Full I-LoRA	71.84	88.44	93.57	-
-Two stage	68.87	85.06	90.48	$\downarrow$3.72
-Setup set	69.76	85.92	93.45	$\downarrow$1.86
-Rand. scale	69.73	87.00	92.77	$\downarrow$1.71

Table 3: Systematic ablation study results. Metrics represent GLUE composite scores. Δ indicates average performance degradation across three tasks compared to full I-LoRA.

To analyze component contributions, we design systematic ablation experiments testing the impact of removing key I-LoRA components during DeBERTaV3-base fine-tuning on CoLA, RTE, and MRPC. As shown in Table 3, compared to full I-LoRA:

- Removing stage separation (omitting retraining after rank setting) causes a 3.72% performance drop.
- Removing the validation set (using training data for both model training and importance scoring in Stage 1) leads to a 1.86% average performance decline, with RTE particularly affected.
- Removing random scaling factors (replacing integrated gradients with instantaneous gradients) results in a 1.71% average performance decrease.

These results validate the importance of all three key components in I-LoRA's architecture.

5.6 Parameter sensitivity analysis

For LoRA rank sensitivity analysis on CoLA dataset (Table 4 and Figure 2), model performance shows non-monotonic variation: Accuracy increases from 71.84 (r=2) to 71.98 (r=16), then drops to 70.79 (r=32). This "bell curve" suggests an optimal rank range balancing model capacity and regularization. I-LoRA demonstrates tighter performance clustering (70.79-71.98) before degradation, verifying improved robustness to rank misestimation through integrated gradient scoring.

LoRA Rank (r)	Accuracy (%)
2	71.84
4	71.72
8	71.98
16	71.98
32	70.79

Table 4: I-LoRA performance on CoLA dataset with varying LoRA ranks

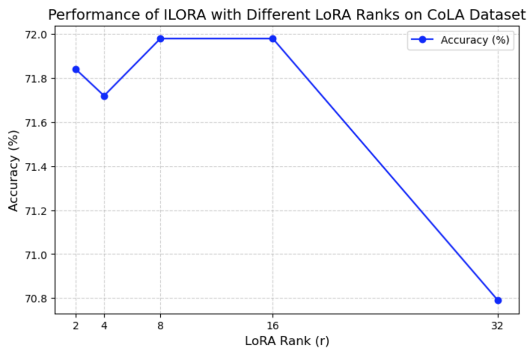

Figure 2: Performance of I-LoRA with different LoRA ranks on CoLA dataset

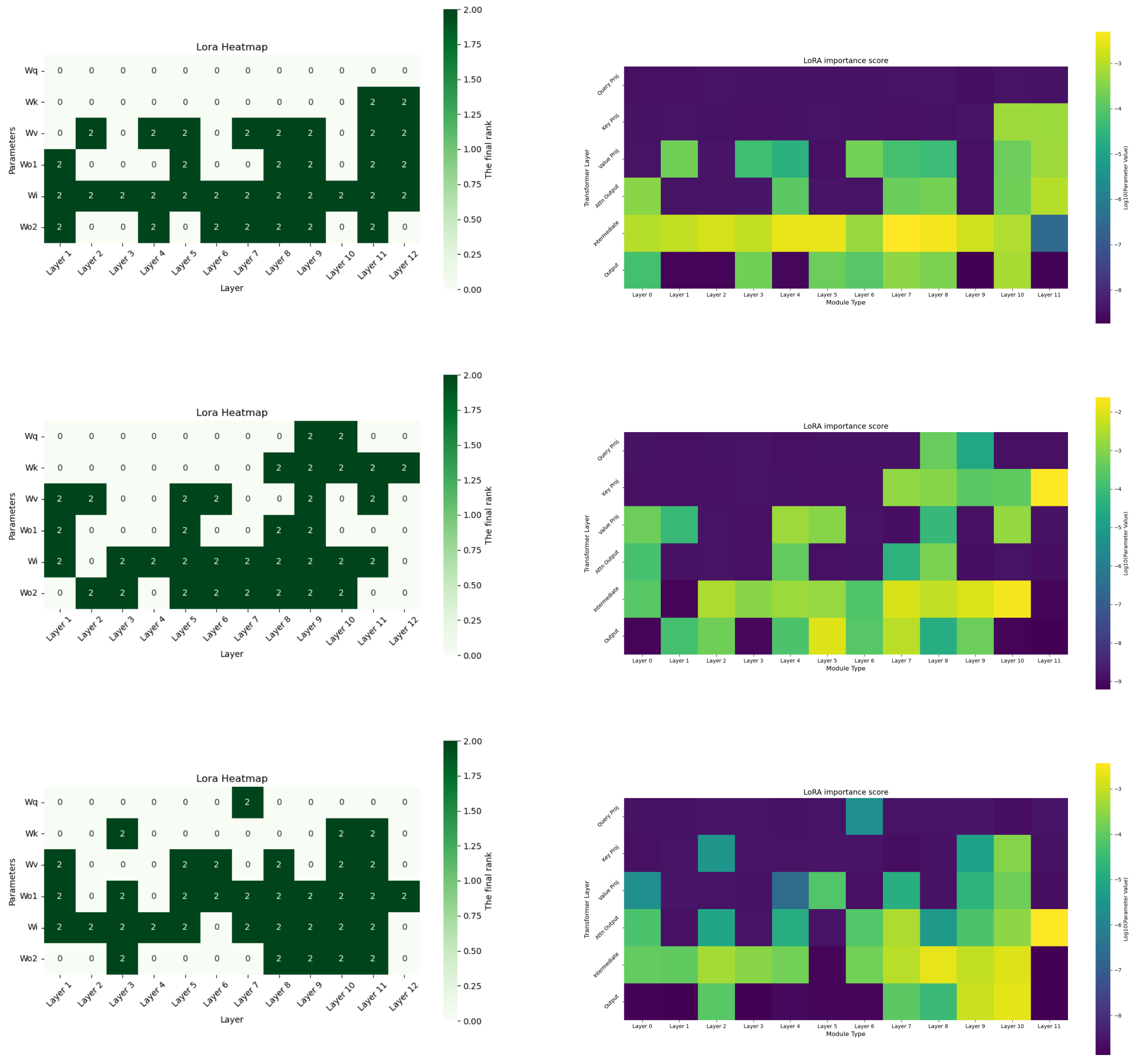

Figure 3: Parameter distribution visualization: Layer-wise rank allocation heatmaps. Each row demonstrates the heatmap of rank allocations across layers (left) and the heatmap of importance scores of each module (right). From top to bottom: heatmaps of fine-tuning DeBERTaV3-base with I-LoRA on CoLA, RTE, and MRPC tasks.

5.7 Parameter distribution and structural analysis

Heatmap visualization (Figure 3) shows different parameter allocation patterns in I-LoRA, linking rank allocation to parameter importance. For DeBERTaV3-base: (a) Higher layers' average rank (8-12) is 3.9, 2.2 times higher than 1.8 of lower layers (1-3), thus consistent with the semantic integration of higher layers. (b) Wi of FFN achieves highest average rank, highlighting its importance.

6 Conclusion

We propose I-LoRA, an interpretable low-rank adaptation framework, which addresses the key limitations of existing LoRA methods through adaptive rank assignment and gradient-based importance scoring. Our two-stage training strategy decouples parameter importance calibration from task-specific tuning, thus alleviating overfitting and enhancing generalization. Integrated gradient is incorporated into the importance score to solve the gradient saturation problem, thus achieving more reliability and interpretability. Extensive experimental results on natural language understanding and text generation tasks show better generalization performance for models fine-tuned using the I-LoRA. In addition, systematic ablation studies provide evidence for the effectiveness of our method.

Limitations

Despite the promising results achieved by I-LoRA, our study has several limitations that warrant discussion. Firstly, due to computational constraints and time limitations, we were unable to leverage the latest state-of-the-art models for our experiments. This omission may restrict the generalizability of our findings to more advanced architectures. Secondly, our evaluation on the XSum dataset was conducted on only one-fifth of the total data due to the high computational demands. This partial evaluation may not fully capture the performance of I-LoRA on larger datasets, potentially limiting the robustness of our conclusions. Future work should address these constraints by incorporating more recent models and conducting comprehensive evaluations on full datasets.

Ethics Statement

This research focuses on the fundamental aspects of large language model fine-tuning, specifically through the development of I-LoRA, a parameter-efficient fine-tuning method. As such, our work does not involve direct application to sensitive domains or data that could raise ethical concerns. We adhere to standard ethical guidelines in machine learning research, ensuring that our methods and experiments are conducted responsibly. Since our work is primarily foundational and does not involve the deployment of models in real-world scenarios, there are no foreseeable ethical risks associated with this study. We commit to transparency and reproducibility, providing detailed methodologies and results to facilitate further research and scrutiny.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. 2023. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4133–4145.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

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

N. Houlsby, S. Bhojanapalli, D. Isayev, and R. Gupta. 2020. Parameter-efficient transfer learning for nlp. *arXiv preprint arXiv:1902.10536*.

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

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

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

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

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020b. 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, Online. Association for Computational Linguistics.

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*.

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.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. arxiv 2019. *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, (721):8026–8037.

A. Radford, S. Narang, T. Salimans, and I. Sutskever. 2018. Improving language understanding by generative pre-training. *arXiv preprint arXiv:1801.05859*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. 2021. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–14.

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

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. 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*, pages 3274–3287.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

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

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32.

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.

Feiyu Zhang, Liangzhi Li, Junhao Chen, Zhouqiang Jiang, Bowen Wang, and Yiming Qian. 2023a. Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning. *arXiv preprint arXiv:2308.12043*.

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

Ruiyi Zhang, Rushi Qiang, Sai Ashish Somayajula, and Pengtao Xie. 2024. 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)*, pages 5048–5060.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

A Dataset Statistics

We present the dataset statistics of the datasets used in our research

B Training Hyper Paramaters

In this section, we present the training hyperparameters used in this research.

B.1 Natural Language Understanding

Following is the training detail of NLP tasks.

Dataset Name	Task	#Train	#Dev	#Test	#Label	Metrics
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr
SST2	Sentiment	67k	872	1.8k	2	Accuracy
MNLI	NLI	393k	20k	20k	3	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
QQP	Paraphrase	364k	40k	391k	2	Accuracy/F1
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy/F1
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman corr

Table 5: Summary of the GLUE benchmark, #Train, #Dev, #Test, and #Label represents the number of samples in each dataset.

B.1.1 GLUE training detail

In our experiment, we tune the learning rate from 4×10^{-4} to 2.2×10^{-3}, and try to find the best learning rate for every method. The batch size for each dataset and each method is all set to 32.

B.1.2 GLUE Extra Parameter Distribution Figures

B.2 Text Generation

Following is the training detail of NLG tasks.

B.2.1 XSum training detail

Dataset Name	learning rate	batch size	#epochs	learning rate	batch size	#epochs
	Rank Setup			**Model Finetune**		
CoLA	4×10^{-4}	32	50	6.9×10^{-4}	32	25
SST2	4×10^{-4}	32	48	4×10^{-4}	32	10
MNLI	4×10^{-4}	32	50	5.1×10^{-4}	32	25
RTE	1.2×10^{-3}	32	55	1.2×10^{-3}	32	50
QQP	4×10^{-4}	32	5	8×10^{-4}	32	15
MRPC	1×10^{-3}	32	30	1×10^{-3}	32	35
QNLI	5×10^{-4}	32	7	5×10^{-4}	32	40
STS-B	2.2×10^{-3}	32	25	2×10^{-3}	32	25

Table 6: Summary of the GLUE training parameters.

Dataset Name	learning rate	batch size	#epochs	learning rate	batch size	#epochs
	Rank Setup			**Model Finetune**		
XSum	2×10^{-4}	24	50	4×10^{-4}	24	50

Table 7: Summary of the XSum training parameters.

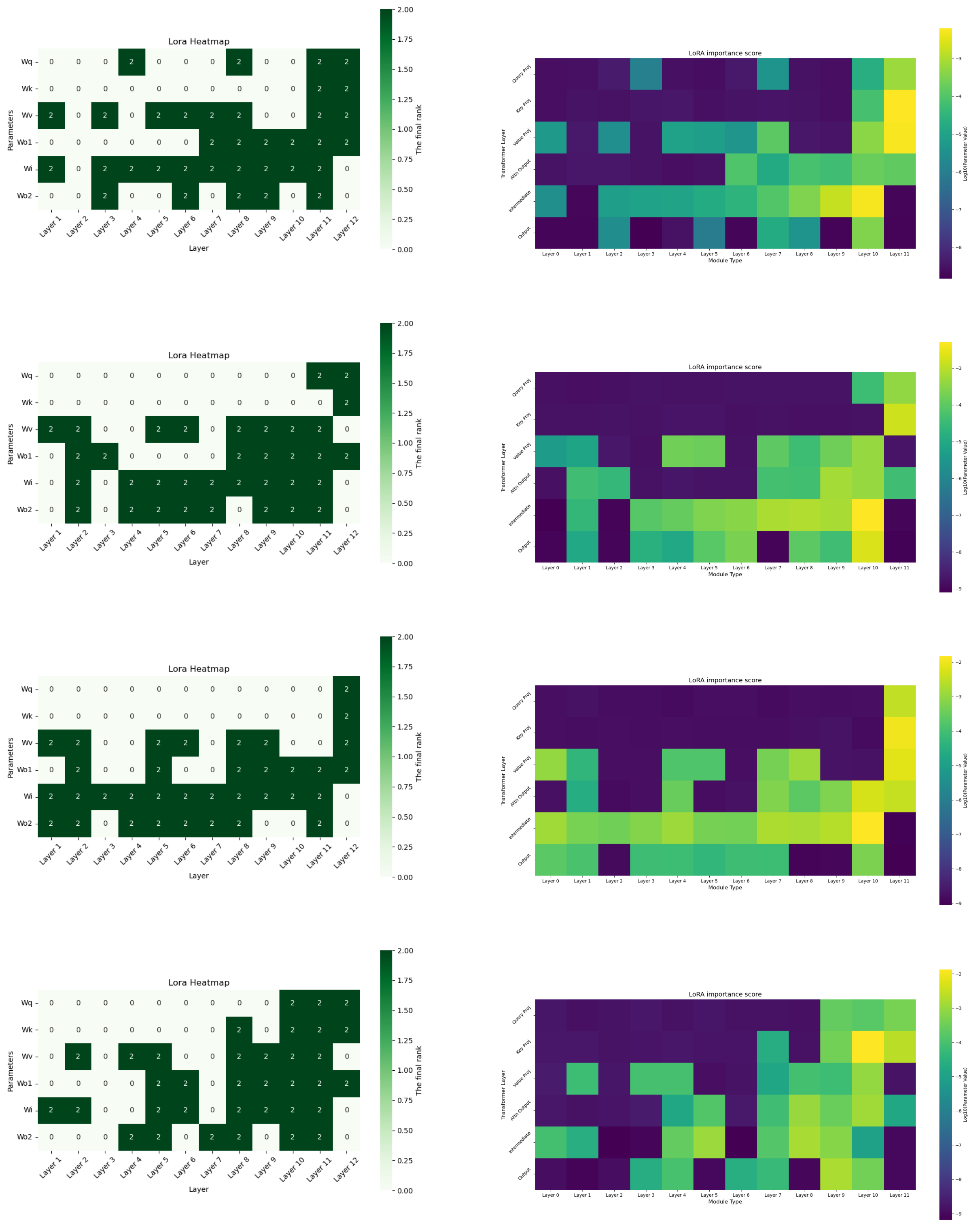

Figure 4: Parameter distribution visualization: Layer-wise rank allocation heatmaps. Each row demonstrates the heatmap of rank allocations across layers (left) and the heatmap of importance scores of each module (right). From top to bottom: heatmaps of fine-tuning DeBERTaV3-base with I-LoRA on qqp, qnli, sst2, stsb tasks.