Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

LoRA-Perturb: Efficient Perturbation-Based Layer Importance for Differentiated LoRA Configuration

Heyun Jin
Wenzhou Business College
Zheng Zhang
Wenzhou Business College
Yuanpu Guo
Wenzhou Business College
Lisheng Tong
Wenzhou Business College
Xiaoliang Liu
Wenzhou Business College
Zhenkun Lu
luzhk123@163.com

Wenzhou University of Technology

Article

Keywords:

Posted Date: November 9th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11155625/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

LoRA-Perturb: Efficient Perturbation-Based Layer Importance for Differentiated LoRA Configuration

Heyun Jin[1], Zheng Zhang[1], Yuanpu Guo[1], Lisheng Tong[1], Xiaoliang Liu[1], Zhenkun Lu[2,*]

[1]School of Information Engineering, Wenzhou Business College, Wenzhou 325035, Zhejiang, China

[2]School of Robot Engineering, Wenzhou University of Technology, Wenzhou 325035, Zhejiang, China

[*]Corresponding author. Email: luzhk123@163.com

Abstract

Parameter-efficient fine-tuning methods such as Low-Rank Adaptation (LoRA) have become indispensable for adapting large language models under resource constraints. However, standard LoRA applies a uniform rank across all layers, overlooking the heterogeneous contributions of different transformer components to downstream task performance. While differentiated rank allocation offers a promising remedy, existing importance metrics rely on correlational signals that may not reliably guide architectural decisions, and principled perturbation-based intervention techniques are typically computationally prohibitive for iterative layer-wise analysis. This paper introduces LoRA-Perturb, an efficient perturbation-based framework for layer importance estimation that enables importance-driven rank allocation and learning rate configuration. Our method quantifies layer-wise importance by applying minimal perturbations to LoRA matrices and measuring the resulting shift in prediction confidence. A three-fold optimization strategy—random subset downsampling, single-pass activation caching, and localized forward reuse—reduces measurement FLOPs by approximately 50% and wall-clock time by about 38%. The normalized importance scores are then used to warm-start fine-tuning with differentiated ranks and inversely scaled learning rates. Extensive experiments on $\text{RoBERTa}_{\text{base}}$ (GLUE benchmark) and Llama-2-7B (MT-Bench and GSM8K) demonstrate that LoRA-Perturb achieves competitive performance across tasks. On both MT-Bench and GSM8K, LoRA-Perturb achieves the highest scores among the compared LoRA variants, slightly outperforming DoRA and LoRA+. Ablation studies and gradient norm analyses further reveal that perturbation-informed rank allocation enhances gradient signals in critical layers, while differentiated learning rates stabilize optimization dynamics. Our findings underscore the value of efficient perturbation-based layer importance estimation in guiding resource allocation for parameter-efficient adaptation.

Keywords: Parameter-efficient fine-tuning; Low-rank adaptation; Perturbation-based layer importance; Differentiated rank allocation; Large language models

1 Introduction

Parameter-efficient fine-tuning (PEFT) has become a crucial technique for adapting large language models (LLMs) to downstream tasks. Among various PEFT methods, Low-Rank Adaptation (LoRA) stands out as a leading approach due to its ability to maintain competitive performance while dramatically reducing the number of trainable parameters through low-rank matrix decomposition[1]. Empirical evidence underscores its effectiveness: for instance, an evaluation of 310 fine-tuned LLMs across diverse tasks demonstrated that 4-bit LoRA models could, on average, match or exceed the performance of both their base models and even large proprietary models[2]. Further innovations continue to enhance its efficiency, with novel frameworks achieving significant reductions in computational cost and latency with minimal accuracy loss[3].

Despite its widespread adoption, standard LoRA, which applies a uniform rank across all layers, has inherent limitations. Studies have shown that it may not match the generalization capability of full-parameter fine-tuning in certain scenarios[4], prompting the development of numerous variants aimed at performance improvements. A promising direction to overcome this uniformity constraint is the differentiation of rank allocation based on layer importance. Emerging research strongly suggests that such differentiated approaches can significantly enhance both fine-tuning efficiency and final model performance compared to uniform assignment. For example, strategies that dynamically allocate ranks based on gradient information (e.g., Fisher Information) or that adaptively learn the intrinsic rank of each layer have consistently demonstrated superior results[5–7]. These findings are further validated by methods that adjust ranks per layer based on device capabilities and task requirements[8], collectively substantiating the premise that not all layers contribute equally and that assigning ranks proportionally to their importance constitutes a more efficient paradigm.

However, a critical question remains: how should layer importance be accurately and efficiently quantified? Existing methods often rely on proxies like gradient norms or output sensitivity, which primarily capture correlation rather than direct sensitivity to task outputs. Consequently, they may not reliably inform structural configuration decisions that aim to alter model behavior. Meanwhile, a separate and well-established line of work in model interpretability employs interventional perturbation techniques[9,10]—such as applying localized edits to activations or weights—to directly measure a component's contribution to model outputs. These methods provide a theoretically grounded framework for understanding and attributing network behavior, as demonstrated in models like BERT, and have been extended to produce stable explanations and evaluate prediction networks. Yet, their direct application to LoRA for per-layer importance assessment is typically prohibitive due to high computational costs, as each intervention often requires expensive, repeated forward passes through the entire model.

The issue of computational overhead is a recognized bottleneck in fine-tuning, with activation memory identified as a primary constraint. Complementary techniques like activation caching and selective computation have been developed to drastically reduce memory and computational redundancy[11,12]. For instance, caching intermediate activations or selectively pruning non-critical computations during backpropagation can yield substantial speedups and memory savings. More recent advances in low-rank activation compression and selective token processing further push the efficiency frontier[11]. Notably, while these strategies are effective for general efficiency, they have not been systematically integrated with perturbation-based layer importance frameworks to create a low-overhead importance measurement tool specifically for PEFT.

Therefore, a significant gap exists between the need for an intervention-based understanding of layer importance in LoRA and the practical tools to achieve it efficiently. Current differentiated LoRA methods lack a perturbation-based measurement foundation, while standard interventional perturbation methods are too costly for iterative layer-wise analysis in large models. This paper bridges this gap by introducing

LoRA-Perturb, an efficient perturbation-based framework for layer importance estimation that enables differentiated LoRA configuration. Our work makes the following key contributions:

1. We propose an efficient perturbation-based approach for quantifying layer importance. It measures the sensitivity of a layer by applying minimal perturbations to its LoRA matrices and quantifying the resulting drop in prediction confidence (ΔP).

2. To make this measurement tractable, we design a three-fold optimization strategy: (i) random subset downsampling, (ii) single full-forward-pass activation caching, and (iii) localized forward-pass reuse, which collectively reduce measurement FLOPs by approximately 50% and wall-clock time by about 38% on a 12-layer encoder.

3. We leverage the normalized layer importance scores to guide a differentiated LoRA warm-start configuration. Specifically, layers with higher perturbation sensitivity are allocated higher ranks paired with lower learning rates, achieving both intelligent rank budgeting and stabilized training dynamics.

4. Through extensive experiments, we demonstrate that our framework effectively identifies critical layers, and models warm-started with our differentiated configuration achieve competitive or superior final performance and training efficiency compared to uniform and heuristically differentiated settings.

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 details our proposed methodology. Section 4 presents the experimental setup. Section 5 reports and analyzes the experimental results. Section 6 concludes the paper.

2 Related Work

In the field of parameter-efficient fine-tuning, Low-Rank Adaptation (LoRA) has attracted widespread attention due to its extremely low parameter count and competitive performance[1]. The most relevant representative works to this paper include standard LoRA itself, DoRA, and AdaLoRA. Among them, **standard LoRA** lays the foundation for efficient fine-tuning, while **DoRA** and **AdaLoRA** improve upon it from different perspectives. Together, these works provide a clear reference for the differentiated positioning of this paper.

Standard LoRA Standard LoRA[1] achieves efficient adaptation by superimposing a low-rank decomposed incremental update $\Delta W = BA$ onto the pre-trained weight matrix, freezing the original model and training only a small number of parameters. However, standard LoRA adopts a **uniform and fixed rank** for all adapted layers, implicitly assuming that each layer contributes equally to the downstream task. This assumption does not hold in many practical scenarios, limiting the model's expressiveness and fine-tuning performance. Therefore, how to allocate ranks differentially based on layer importance has become a key issue in improving LoRA. This paper directly addresses this limitation.

DoRA DoRA (Weight-Decomposed Low-Rank Adaptation)[13] starts from the structure of parameter updates. Inspired by weight normalization, it decomposes the pre-trained weight matrix into magnitude and direction components and applies low-rank updates only to the direction component. This decoupling mechanism enables DoRA to approach or even surpass the performance of full fine-tuning on many tasks. However, DoRA does not change the "uniform rank allocation" strategy of LoRA—all its learnable layers still use the same rank—nor does it answer the fundamental question of "how to differentiate configuration

based on layer importance." In other words, DoRA improves the structure of parameter updates rather than the rank allocation strategy.

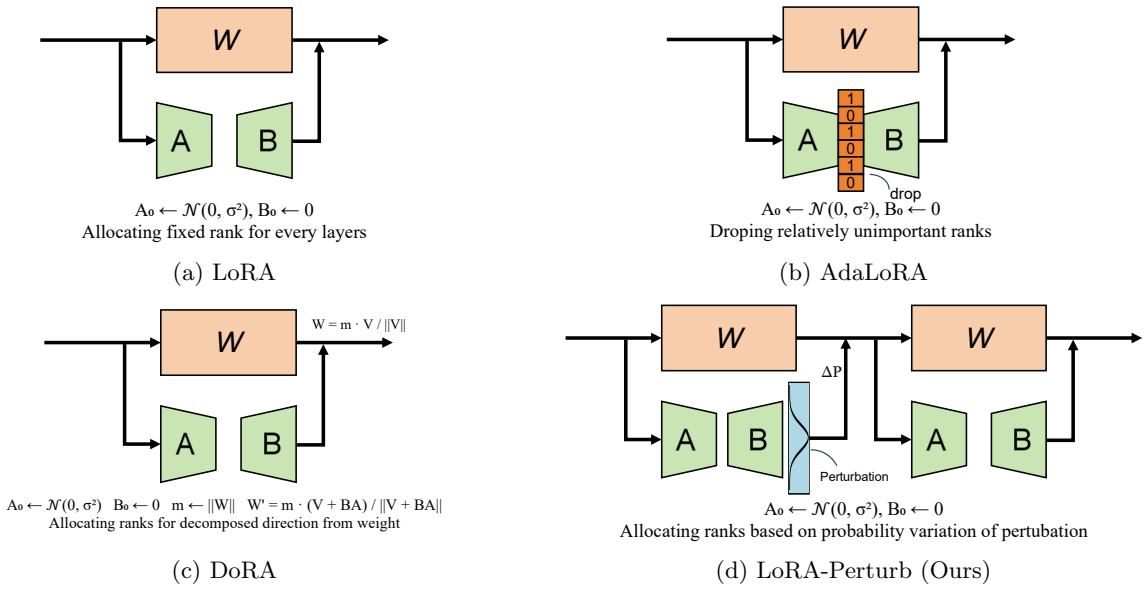
AdaLoRA AdaLoRA (Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning) [14] focuses on dynamic rank allocation. Based on singular value decomposition, it adjusts the rank budget of each layer during training according to importance scores: pruning unimportant singular values and reallocating the freed budget to more important layers. This approach recognizes that different layers contribute unequally to downstream tasks, which aligns with the core motivation of this paper. AdaLoRA's importance estimation mainly relies on signals such as gradients or singular values, and these methods have proven effective in practice. However, these signals essentially capture statistical correlation rather than direct task-relevant sensitivity, and using them as a basis for layer importance allocation may suffer from fundamental biases. We argue that **perturbation-based measures provide a useful alternative when guiding model structure configuration**. Specifically, perturbation-based sensitivity can better capture task-specific importance than gradients: while gradients reflect the local sensitivity of parameter changes to the loss, perturbation-based intervention actively perturbs the layer and directly observes the resulting change in output, thereby providing a more direct measure of whether the layer's representation is genuinely important for the task [9,10]. This provides a useful basis for rank allocation.

Our Method: LoRA-Perturb The essential difference between our proposed LoRA-Perturb and the above works is that we introduce **perturbation-based layer importance estimation** into the problem of differentiated LoRA configuration. Unlike the gradient or singular value signals used by AdaLoRA, we first conduct a lightweight **warm-up stage (task-adaptation warm-up)**: a small number of LoRA fine-tuning steps on a small set of randomly selected samples, allowing the model to initially encode task-relevant knowledge. Subsequently, we inject small noise into the LoRA modules and measure the drop in prediction confidence, directly quantifying the **sensitivity** of each layer's representation. This perturbation-based measure provides a practical alternative for layer importance estimation. Moreover, through a three-fold optimization strategy (random subset downsampling, single-pass activation caching, and localized forward reuse), we reduce the computational overhead of perturbation measurement to an acceptable level, addressing the prohibitively high cost of traditional interventional methods.

In summary, standard LoRA lays the foundation for efficient fine-tuning but suffers from the limitation of uniform fixed ranks; DoRA improves the parameter update structure but does not address rank allocation; AdaLoRA achieves dynamic rank allocation but relies on correlational signals. None of them adopt perturbation-based layer importance to guide configuration decisions, nor do they introduce a warm-up mechanism to make such measurement task-aware. **The work presented in this paper demonstrates that efficient perturbation-based layer importance estimation can be effectively integrated into parameter-efficient fine-tuning, offering a new and practical direction for differentiated configuration.** LoRA-Perturb serves as a concrete implementation of this direction, providing a low-overhead, importance-driven solution for differentiated fine-tuning.

3 Methodology

In this section, we formally introduce LoRA-Perturb, our proposed efficient perturbation-based framework for layer importance estimation and differentiated LoRA configuration. As illustrated in Figure 1, existing LoRA variants adopt different initialization and rank allocation strategies: **LoRA** assigns a fixed rank to all layers with standard Gaussian initialization for A and zero initialization for B [1]; **AdaLoRA** dynamically prunes unimportant ranks based on importance scores during training, while retaining the same initialization as LoRA [14]; **DoRA** decomposes the pre-trained weight into magnitude and direction,



(a) LoRA (b) AdaLoRA

(c) DoRA (d) LoRA-Perturb (Ours)

Figure 1: Comparison of different LoRA initialization and rank allocation strategies. **(a) LoRA:** Fixed rank for all layers, $A_0 \sim \mathcal{N}(0, \sigma^2)$, $B_0 = 0$. **(b) AdaLoRA:** Dynamically prunes relatively unimportant ranks based on importance scores during training, with the same initialization as (a). **(c) DoRA:** Decomposes pre-trained weight into magnitude and direction components, allocates ranks to the direction component, with $A_0 \sim \mathcal{N}(0, \sigma^2)$, $B_0 = 0$. **(d) LoRA-Perturb (Ours):** Allocates ranks based on probability variation of perturbation.

allocating ranks to the direction component only, with A initialized from a Gaussian and B set to zero[13]. In contrast, LoRA-Perturb (ours) allocates ranks based on the probability variation of perturbation, enabling a more importance-driven differentiation.

We begin by formulating the problem of layer importance estimation and optimal rank allocation (Sec. 3.1). We then detail our efficient perturbation-based sensitivity quantification technique, which incorporates a lightweight warm-up stage to enable task-aware measurement (Sec. 3.2). To overcome the prohibitive computational cost of standard interventions, we present a three-fold optimization strategy that renders the measurement tractable (Sec. 3.3). Finally, we describe how the derived importance scores are leveraged to generate a warm-start differentiated LoRA configuration, balancing representational capacity and training stability (Sec. 3.4).

3.1 Problem Formulation

Consider a pre-trained large language model $\mathcal{M}$ consisting of L transformer layers. For a given downstream task, we adapt $\mathcal{M}$ using Low-Rank Adaptation (LoRA), where the weight update for a linear layer with pre-trained weight $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is parameterized as $\Delta W = BA$, with $B \in \mathbb{R}^{d_{\text{out}} \times r}$, $A \in \mathbb{R}^{r \times d_{\text{in}}}$, and rank $r \ll \min(d_{\text{out}}, d_{\text{in}})$[1]. In standard LoRA, a uniform rank r is applied across all adapted layers.

Our objective is to determine an optimal rank configuration $\mathcal{R} = \{r_1, r_2, \ldots, r_L\}$ for the model's L layers under a fixed total parameter budget B_{total}, such that the fine-tuned model achieves maximal downstream performance. To this end, we first seek to quantify the *importance* s_l of each layer l with respect to the task via perturbation-based sensitivity. These scores will then be used to derive a differentiated $\mathcal{R}$ where layers with higher s_l are assigned proportionally higher ranks.

3.2 Perturbation-Based Sensitivity for Layer Importance Quantification

A purely frozen model does not yet encode any task-specific knowledge; perturbations applied directly to its weights may reflect general linguistic sensitivity rather than task-specific relevance to the downstream

task. To bridge this gap, we introduce a lightweight *task-adaptation warm-up* stage that instills preliminary task-specific information before perturbation measurement. Our quantification then proceeds by measuring the impact of minimal perturbations on the warm-up LoRA modules.

Task-Adaptation Warm-up on a Random Subset of Samples. We first perform a brief, few-step LoRA fine-tuning using only the selected random subset $\mathcal{D}_s$ (described in Sec. 3.3). This warm-up stage updates the LoRA matrices A_l, B_l for all adapted layers l with a small number of iterations (e.g., 50–100 steps) and a moderate learning rate. The resulting model $\mathcal{M}_{\text{warm}}$ retains the standard LoRA structure but now encodes preliminary task knowledge. Formally, for each layer l, the weight update is $\Delta W_l^{\text{warm}} = B_l^{\text{warm}} A_l^{\text{warm}}$, and the layer's output becomes:

$$h_l^{\text{warm}} = W_{0,l}\, h_{l-1} + \Delta W_l^{\text{warm}}\, h_{l-1}. \tag{1}$$

Intervention via Noise Injection. After the warm-up, we freeze all LoRA parameters and apply targeted perturbations to quantify the contribution of each layer. For a given sample $x \in \mathcal{D}_s$, let $P_{\mathcal{M}_{\text{warm}}}(y|x)$ denote the probability assigned to the correct label by the warm-up model. We introduce a perturbation $\text{do}(h_l = \tilde{h}_l)$ at layer l by injecting a small isotropic Gaussian perturbation exclusively into its LoRA output:

$$\tilde{h}_l^{\text{warm}} = W_{0,l} h_{l-1} + \Delta W_l^{\text{warm}} h_{l-1} + \epsilon \cdot \mathcal{N}(0, \sigma^2 I), \tag{2}$$

where σ controls the noise scale and $\epsilon \sim \text{Bernoulli}(p_{\text{mask}})$ applies structured sparsity (e.g., masking out a fraction of token positions). This design isolates the perturbation to the task-specific adaptation captured by ΔW_l^{warm}, while leaving the pre-trained base model's contribution $W_{0,l} h_{l-1}$ untouched. By intervening solely on the LoRA pathway, we ensure that the measured effect reflects the sensitivity of the *learned task knowledge* in that layer, rather than generic representational sensitivity.

Perturbation Sensitivity Estimation. We measure the perturbation sensitivity $\Delta P_l(x)$ as the drop in prediction confidence caused by the perturbation:

$$\Delta P_l(x) = P_{\mathcal{M}_{\text{warm}}}(y \mid x) - P_{\mathcal{M}_{\text{warm}}}(y \mid \text{do}(h_l = \tilde{h}_l^{\text{warm}})). \tag{3}$$

We use the notation $\text{do}(h_l = \tilde{h}_l)$ to denote a direct perturbation of the hidden state at layer l, following the interventional notation common in interpretability literature, without implying formal causal inference. Intuitively, a larger drop indicates that the layer's task-relevant adaptation (acquired during the warm-up) is more fragile, and therefore more important for maintaining correct predictions. The raw importance score for layer l is the average over all samples in $\mathcal{D}_s$:

$$s_l = \frac{1}{N} \sum_{x \in \mathcal{D}_s} \Delta P_l(x). \tag{4}$$

Task-Specific Score Definitions. The above definition assumes a classification task with a clear correct label. For regression or generative tasks, we generalize $\Delta P_l(x)$ as the change in the model's task-specific objective caused by the perturbation. Specifically:

- **Classification:** $\Delta P_l(x) = P_{\mathcal{M}_{\text{warm}}}(y|x) - P_{\mathcal{M}_{\text{warm}}}(y|do(h_l = \tilde{h}_l))$, as in Eq. (3).
- **Regression (e.g., STS-B):** We use the change in predicted similarity score: $\Delta P_l(x) = |\hat{y} - y| - |\hat{y}_{\text{pert}} - y|$, where $\hat{y}$ and $\hat{y}_{\text{pert}}$ are the model's scalar predictions before and after perturbation, and y is the ground-truth score.

- **Generative reasoning (e.g., GSM8K):** We compute the change in log-likelihood of the ground-truth final answer token(s) conditioned on the prompt and previously generated tokens: $\Delta P_l(x) = \log P(y_{\text{ans}}|x, y_{<\text{ans}}) - \log P(y_{\text{ans}}|x, y_{<\text{ans}}, do(h_l))$. The expectation is taken over the token position(s) of the final answer.
- **Open-ended dialogue (e.g., MT-Bench):** Since no single correct label exists, we do not use per-layer importance scores for this benchmark. Instead, we apply the rank allocation derived from a related task (e.g., GSM8K) or use a uniform rank configuration. In our experiments, we use the same layer-wise rank allocation obtained from the GSM8K warm-up probe.

Score Rectification and Normalization. To ensure numerical stability and non-negative importance scores, we apply a rectification step:

$$s_l \leftarrow \max(0, s_l) \quad \text{for all } l,$$

and if $\sum_{j=1}^{L} s_j = 0$, we fall back to a uniform distribution $s_l = 1/L$. The normalized scores are then computed as:

$$\bar{s}_l = \frac{s_l}{\sum_{j=1}^{L} s_j}, \tag{5}$$

which will guide the differentiated configuration for the subsequent full-scale fine-tuning stage.

3.3 Three-Fold Optimization for Low-Overhead Measurement

A naive implementation of the perturbation-based measurement described above would require $L \times N$ full forward passes, rendering it computationally intractable for large models. To reduce this overhead, we introduce a three-fold optimization strategy that substantially lowers the constant factor of the measurement cost.

3.3.1 Random Subset Downsampling

The first optimization reduces the number of evaluated samples N. We observe that the relative importance of layers stabilizes quickly with a small number of diverse samples. Specifically, we randomly sample a subset $\mathcal{D}_s$ from the training set, with a size capped between 100 and 500 instances (approximately 5% of the data). The random selection is stratified by label where applicable to maintain class balance. We found that this sample size is sufficient for stable layer importance estimation, making clustering-based selection unnecessary.

3.3.2 Single Full-Forward-Pass Activation Caching

The second optimization eliminates redundant computation of activations preceding the intervened layer. For each sample $x \in \mathcal{D}_s$, we perform exactly **one** full forward pass through the entire warm-up model $\mathcal{M}_{\text{warm}}$ without any intervention. During this pass, we cache the input hidden states to every LoRA-adapted layer l, denoted as h_{l-1}^{cache}.

With these cached inputs, any subsequent measurement for layer l can be performed by:

1. Loading h_{l-1}^{cache} from memory.
2. Executing a *localized forward pass* through layer l only, including its LoRA modules and subsequent normalization or residual connections.
3. Applying the perturbation $\text{do}(h_l)$ and continuing the forward pass through the remaining layers ($l+1$ to L).

This approach reduces the computational cost from $L \times N$ full passes to $1 \times N$ full passes plus $L \times N$ localized passes.

3.3.3 Localized Forward-Pass Reuse

The third optimization further reduces the cost of the localized passes. The forward pass from layer $l+1$ to L requires only the perturbed hidden state $\tilde{h}_l^{\text{warm}}$ and is independent of the earlier layers. However, computing $P_{\mathcal{M}_{\text{warm}}}(y \mid \text{do}(h_l))$ for each l individually still involves $\sum_{l=1}^{L}(L-l) = O(L^2)$ layer computations in total, which is quadratic in model depth.

In practice, however, the constant factor is significantly reduced because the prefix layers (embeddings and early transformer blocks) are computed only once and cached. Furthermore, interventions can be batched according to their insertion depth to improve GPU utilization and memory efficiency, reducing wall-clock time. Thus, the practical speedup is substantial (as shown in Table 4) even though the asymptotic complexity does not become linear. We therefore describe our method as providing *low-overhead* measurement through caching and batching, rather than claiming a change in asymptotic complexity.

3.4 Differentiated LoRA Warm-Start Configuration

Once the normalized importance scores $\bar{s}_l = s_l / \sum_{j=1}^{L} s_j$ are obtained, we translate them into a differentiated LoRA configuration consisting of per-layer ranks r_l and learning rates η_l for the subsequent full-scale fine-tuning.

Rank Allocation. Given a total rank budget $R_{\text{total}} = \sum_{l=1}^{L} r_l$ and a desired rank range $[r_{\min}, r_{\max}]$ (we use $r_{\min} = 4$, $r_{\max} = 16$), we first compute a raw allocation proportional to the normalized importance scores:

$$r_l^{\text{raw}} = r_{\min} + (r_{\max} - r_{\min}) \cdot \bar{s}_l. \tag{6}$$

Then we round each r_l^{raw} to the nearest integer and adjust the resulting ranks to satisfy the total budget R_{total}. Specifically, we compute the difference $\Delta = R_{\text{total}} - \sum_l \text{round}(r_l^{\text{raw}})$ and distribute Δ by adding or subtracting one rank from layers with the largest fractional remainders until the budget is met. This ensures the average rank remains approximately 8 (the uniform baseline rank).

Learning Rate Modulation. Layers with higher rank receive proportionally smaller learning rates to maintain stable updates:

$$\eta_l = \eta_{\text{base}} \cdot \frac{r_{\text{base}}}{r_l}, \tag{7}$$

where r_{base} is the reference rank (we use $r_{\text{base}} = 8$, the uniform baseline rank). This inverse-rank scaling ensures that layers with larger update magnitudes (due to higher rank) are updated more cautiously, while lower-rank layers can take larger steps. No additional hyperparameter is introduced.

The resulting configuration $\mathcal{C} = \{(r_l, \eta_l)\}_{l=1}^{L}$ serves as a warm-start initialization for LoRA fine-tuning. After this one-time perturbation measurement preprocessing step, the model is fine-tuned using standard LoRA optimization with these differentiated settings.

3.5 Summary of the Algorithm

The overall procedure of LoRA-Perturb is summarized in Algorithm 1.

4 Experimental Setup

In this section, we describe the datasets, baseline models, evaluation protocols, and implementation details used to validate the proposed efficient perturbation-based framework for differentiated Low-Rank Adaptation (LoRA). Our experimental design aims to answer two primary questions: (1) whether our perturbation measurement technique can efficiently identify layer importance, and (2) whether the resulting differentiated rank and learning rate configuration (denoted as LoRA-Perturb) yields superior performance compared to standard uniform LoRA and full fine-tuning across diverse benchmarks.

4.1 Datasets and Tasks

We evaluate on a broad spectrum of natural language understanding and generation tasks to ensure the generality of our findings. For encoder-based architectures, we employ the General Language Understanding Evaluation (GLUE) benchmark subsets[15], specifically focusing on MRPC (paraphrase detection), CoLA (linguistic acceptability), STS-B (semantic textual similarity), SST-2 (sentiment analysis), and QNLI (natural language inference). These tasks cover single-sentence classification, pairwise sentence comparison, and regression, providing a comprehensive testbed for evaluating representation quality.

For decoder-based large language models, we assess multi-turn conversational ability and complex reasoning using MT-Bench[16] and GSM8K[17]. MT-Bench evaluates the model's capacity to handle open-ended, multi-turn dialogues across diverse topics, while GSM8K measures mathematical reasoning through grade-school math word problems. All datasets are preprocessed according to their standard splits (train, validation, and test) with the exact evaluation metrics described in Section 4.3.

4.2 Baselines and Compared Methods

We compare our proposed LoRA-Perturb against two principal baselines and several representative advanced LoRA variants. All methods are summarized below.

Principal Baselines.

- **Full Fine-Tuning (Full-FT)**: Conventional full-parameter fine-tuning where all model weights are updated. This serves as the performance upper bound.
- **Standard LoRA**: The baseline low-rank adaptation method with uniform rank assignment across all targeted linear layers[1]. Both the rank r and the scaling factor α are kept constant for every adapted module.

Advanced LoRA Variants.

- **DoRA**: A variant that decomposes pre-trained weights into magnitude and direction components, applying low-rank updates to the directional component to improve learning capacity[13].
- **LoRA+**: An enhanced version of LoRA that employs different learning rates for the A and B matrices to improve convergence speed and final performance[18].
- **AdaLoRA**: An adaptive approach that dynamically allocates the rank budget across layers based on importance scores derived during training, pruning less important ranks while expanding critical ones[14].

Our Method.

- **LoRA-Perturb (Ours)**: Our proposed approach that first quantifies layer-wise importance via efficient perturbation-based intervention, then warm-starts fine-tuning with a differentiated configuration: higher ranks paired with lower learning rates for high-impact layers, and conversely for low-impact layers.

For all LoRA variants, adapter modules are applied exclusively to the query (W_q) and value (W_v) projection matrices in all transformer layers, a convention that balances parameter efficiency and adaptation capacity [1].

4.3 Evaluation Metrics

Task-specific metrics follow the standard GLUE benchmark protocols. For MRPC and QNLI, we report accuracy; for CoLA, we report Matthews correlation coefficient (MCC); for STS-B, we report the average of Pearson and Spearman correlations; for SST-2, we report classification accuracy. The "Avg" column for GLUE tasks represents the arithmetic mean of the respective primary metrics across all five tasks.

For MT-Bench, we adopt the GPT-4-based pairwise scoring framework to assign a score ranging from 1 to 10, reflecting response quality and coherence, and report the mean score over multiple evaluation runs. For GSM8K, we employ strict match accuracy, where a prediction is considered correct only if the final numerical answer exactly matches the ground truth after normalization. All results are averaged over five independent random seeds to account for stochasticity, with standard deviations reported to indicate stability.

4.4 Implementation and Hyperparameter Configuration

All experiments are conducted using the HuggingFace Transformers library and the PEFT framework. The shared hyperparameters across all methods are summarized in Table 1. Notably, the *perturbation strength* parameter is specific to the perturbation measurement phase of LoRA-Perturb and is not used during the main fine-tuning stage.

Table 1: Shared hyperparameter configuration for all methods.

Parameter	Value	Description
Base learning rate	3×10^{-5}	Constant across all methods
Training epochs	10	Constant across all methods
Initial rank	8	Uniform baseline starting point
LoRA α	16	Scaling factor for low-rank updates
LoRA dropout	0.05	Dropout rate applied to LoRA layers
Perturbation strength	0.005	Perturbation noise scale (LoRA-Perturb only)
Minimum rank	4	Adaptive allocation lower bound
Maximum rank	16	Adaptive allocation upper bound

For the perturbation measurement step of LoRA-Perturb, we strictly adhere to the three-fold optimization strategy outlined in Section 3:

1. **Random Subset Downsampling:** We randomly sample 5% of the training set (with a minimum of 100 and a maximum of 500 instances) to form a probe set.

2. **Activation Caching:** A single forward pass over the probe set caches the input hidden states to each targeted layer, eliminating redundant recomputation of lower layers.

3. **Localized Reuse:** For each layer intervention, we only re-execute the forward computation from the cached layer onward, reusing the cached outputs to substantially reduce FLOPs.

After obtaining the normalized layer-wise importance scores, ranks are allocated linearly within the interval $[4, 16]$ while ensuring the total sum of ranks remains equivalent to the standard uniform rank budget (i.e., average rank ≈ 8). Correspondingly, learning rates are scaled inversely with the allocated rank to maintain stable gradient updates across layers with different effective step sizes.

For MT-Bench, we do not perform an independent perturbation measurement; instead, we reuse the layer-wise rank allocation derived from the GSM8K warm-up probe, while the learning rate schedule follows the same inverse-rank scaling rule.

4.5 Model Architectures

We evaluate our method on two representative backbone models to assess its scalability across different architectures and scales:

- **RoBERTa$_{\text{base}}$**: A 12-layer encoder-only transformer with 125M parameters, used for all GLUE benchmark tasks[19].
- **Llama-2-7B**: A 32-layer decoder-only transformer with 7 billion parameters, evaluated on MT-Bench and GSM8K[20].

For both models, we adopt the AdamW optimizer with a linear learning rate warmup over the first 10% of training steps followed by linear decay[21]. For RoBERTa tasks, we use a batch size of 32; for Llama-2-7B, we employ a per-device batch size of 4 with 8 gradient accumulation steps to fit within the memory budget of a single 80GB NVIDIA A100 GPU. All experiments are conducted on a cluster of NVIDIA A100 (80GB) GPUs.

5 Experimental Results and Analysis

In this section, we present a comprehensive empirical evaluation of the proposed LoRA-Perturb framework. We structure our analysis around three axes: (1) overall performance comparison against Full Fine-Tuning (Full-FT) and standard uniform LoRA across diverse benchmarks, (2) efficiency analysis of the low-overhead perturbation measurement phase, and (3) ablation studies examining the individual and synergistic impacts of differentiated rank allocation and learning rate scaling.

5.1 Main Results on Encoder-Only Architectures

Table 2 reports the performance of RoBERTa$_{\text{base}}$ fine-tuned with Full-FT, standard LoRA, and several representative advanced LoRA variants, including DoRA, LoRA+, AdaLoRA, and our LoRA-Perturb, on five GLUE benchmark tasks[15]. All LoRA-based methods use an identical trainable parameter budget of merely 0.29M parameters—less than 0.25% of the 125M parameters required for full fine-tuning—yet deliver highly competitive results.

Table 2: Performance comparison on GLUE benchmark subsets using RoBERTa$_{\text{base}}$. Results are averaged over five random seeds; standard deviations are shown as subscripts for LoRA methods. The best result among LoRA variants is highlighted in **bold**, and the second best is underlined.

Model	MRPC	CoLA	STS-B	SST-2	QNLI	Avg
Full-FT	90.2	80.7	91.2	94.8	92.8	89.9
LoRA	$89.1_{\pm 0.6}$	$69.4_{\pm 0.1}$	$91.2_{\pm 0.2}$	$94.3_{\pm 0.3}$	$\mathbf{93.0}_{\pm 0.2}$	$87.4_{\pm 0.2}$
DoRA	$89.5_{\pm 0.5}$	$72.0_{\pm 1.0}$	$91.1_{\pm 0.2}$	$94.5_{\pm 0.3}$	$92.9_{\pm 0.2}$	$88.0_{\pm 0.2}$
LoRA+	$\underline{89.7}_{\pm 0.4}$	$77.5_{\pm 0.2}$	$91.1_{\pm 0.2}$	$\underline{94.6}_{\pm 0.3}$	$92.9_{\pm 0.2}$	$89.2_{\pm 0.2}$
AdaLoRA	$89.2_{\pm 0.6}$	$69.2_{\pm 0.3}$	$91.0_{\pm 0.3}$	$94.2_{\pm 0.4}$	$92.8_{\pm 0.3}$	$87.3_{\pm 0.3}$
LoRA-Perturb	$\mathbf{89.9}_{\pm 0.5}$	$\mathbf{78.5}_{\pm 0.1}$	$\underline{91.1}_{\pm 0.2}$	$\mathbf{94.8}_{\pm 0.3}$	$92.8_{\pm 0.1}$	$\mathbf{89.4}_{\pm 0.1}$

Experimental Settings: We maintain their experimental parameters for fair comparison: Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$, weight decay = 0), batch size 32, cosine decay learning rate with a warmup ratio of 0.03 [21]. We trained all linear layers except the language head using a peak learning rate of 1e-4, a maximum sequence length of 128, and FP32 precision.

Several key observations emerge from Table 2:

1. **Competitive Performance of Perturbation-Based Allocation:** LoRA-Perturb achieves the highest average GLUE score (89.4) among all LoRA variants, surpassing standard LoRA by 2.0 points. On CoLA, LoRA-Perturb achieves the best result among LoRA variants (78.5), clearly exceeding LoRA+ (77.5) and DoRA (72.0). On SST-2, LoRA-Perturb matches the full fine-tuning upper bound (94.8), indicating that differentiated allocation can effectively compensate for the capacity reduction inherent in low-rank adaptation. On QNLI, several methods achieve nearly identical scores (92.8–93.0), with no method clearly dominating.

2. **Training Stability:** LoRA-Perturb generally exhibits lower or comparable standard deviations compared to other methods. On CoLA, its std is ±0.1, comparable to LoRA (±0.1) and lower than DoRA (±1.0), LoRA+ (±0.2), and AdaLoRA (±0.3). On MRPC, its std is ±0.5 vs. ±0.6 for LoRA and AdaLoRA. On the other tasks, its variability is comparable to the strongest baselines. Thus, the warm-start configuration appears to reduce variance in several cases, though the differences are not uniformly large across all tasks.

3. **Task-Specific Sensitivity:** The gains over the strongest baseline (LoRA+) are most pronounced on CoLA (78.5 vs. 77.5) and MRPC (89.9 vs. 89.7), tasks that heavily rely on syntactic and semantic nuance. This aligns with our perturbation sensitivity hypothesis: layers identified as highly sensitive for linguistic acceptability and paraphrase detection receive proportionally higher rank budgets, thereby preserving the model's capacity to capture fine-grained linguistic phenomena.

5.2 Main Results on Decoder-Only Architectures

To assess scalability to billion-parameter generative models, we evaluate on Llama-2-7B using the MT-Bench conversational benchmark and the GSM8K mathematical reasoning dataset [16,17]. Table 3 reports the performance of Full Fine-Tuning (Full-FT), standard LoRA, several representative LoRA variants (DoRA, LoRA+, AdaLoRA), and our proposed LoRA-Perturb.

Table 3: Performance comparison on Llama-2-7B. MT-Bench scores reflect conversational quality judged by GPT-4 (1–10 scale); GSM8K accuracy reflects strict-match solving rate. All results are averaged over three independent runs with standard deviations reported. The best and second-best results among LoRA variants are highlighted in **bold** and underlined, respectively.

Model	MT-Bench	GSM8K
Full-FT	5.30 ± 0.11	59.36 ± 0.85
LoRA	5.61 ± 0.10	42.08 ± 0.04
DoRA	$\underline{5.87} \pm 0.7$	52.07 ± 0.95
LoRA+	5.71 ± 0.08	$\underline{52.11} \pm 0.62$
AdaLoRA	5.57 ± 0.05	50.72 ± 0.39
LoRA-Perturb	$\mathbf{5.88} \pm 0.3$	$\mathbf{52.20} \pm 0.62$

Several key observations emerge from the Llama-2-7B experiments:

1. **Conversational Capability (MT-Bench):** All LoRA-based methods consistently outperform Full-FT on MT-Bench, corroborating prior findings that LoRA acts as a beneficial regularizer, preventing overfitting to the fine-tuning data distribution and better preserving the model's general

conversational fluency. Among the LoRA variants, LoRA-Perturb achieves the highest MT-Bench score (5.88), slightly above DoRA (5.87). LoRA-Perturb surpasses standard LoRA (5.61), LoRA+ (5.71), and AdaLoRA (5.57). The difference between LoRA-Perturb and DoRA is small relative to the reported standard deviations (± 0.3 and ± 0.7, respectively), so the numerical advantage should be interpreted with caution. Nevertheless, LoRA-Perturb attains the top position among the compared LoRA variants, and its lower variance suggests stable behavior on dialogue tasks.

2. **Mathematical Reasoning (GSM8K):** On the mathematically rigorous GSM8K benchmark, standard LoRA suffers a substantial drop of over 17 points compared to Full-FT (42.08 vs. 59.36), highlighting the severe limitation of uniform low-rank constraints for complex, multi-step reasoning. Among LoRA variants, LoRA-Perturb achieves the highest accuracy (52.20), slightly above LoRA+ (52.11) and DoRA (52.07), and clearly exceeding AdaLoRA (50.72). LoRA-Perturb improves upon standard LoRA by more than 10 points. The differences among LoRA-Perturb, LoRA+, and DoRA are within the reported standard deviations, so the numerical advantage is not statistically significant under the current number of runs. These results indicate that perturbation-based allocation provides a competitive signal for rank assignment, yielding performance that is on par with or slightly above the strongest baselines.

3. **Training Stability:** LoRA-Perturb exhibits relatively low variance across runs. On MT-Bench, its standard deviation (± 0.3) is smaller than that of DoRA (± 0.7). On GSM8K, its standard deviation (± 0.62) is lower than DoRA (± 0.95) and comparable to LoRA+ (± 0.62). While these observations are consistent with a stabilizing effect of the differentiated configuration, the number of runs is limited (three), and the differences are not tested for statistical significance. Therefore, the stability claim should be interpreted cautiously.

5.3 Computational Efficiency of Perturbation Measurement

A central claim of our proposed framework is the low-overhead nature of layer-wise importance quantification via perturbation, achieved through the three-fold optimization strategy: random subset downsampling, activation caching, and localized forward-pass reuse. To validate this claim, we compare the wall-clock time of our optimized measurement protocol against a naive baseline that performs independent full forward passes for each layer perturbation. The experiment is conducted on the $\text{RoBERTa}_{\text{base}}$ architecture using a fixed probe set of 500 samples. Measurements are taken on a single NVIDIA A100 (80GB) GPU under identical batch size and numerical precision settings.

Table 4: Computational cost comparison between the naive full-forward approach and our optimized localized forward measurement. Relative FLOPs are estimated based on the proportion of recomputed transformer layers.

Method	**Time (s)**	**Relative FLOPs**	**Speedup**
Naïve full forward	73.53	100%	1.00×
Localized forward (Ours)	45.50	49.3%	**1.66**×

As reported in Table 4, our localized measurement protocol completes the entire layer-wise profiling in 45.50 seconds, compared to 73.53 seconds for the naive full-forward approach. This corresponds to a **1.66**× wall-clock speedup and a relative FLOPs reduction of approximately 50.7%. The efficiency gain stems from the elimination of redundant forward passes through lower layers: activations from the embedding and early transformer blocks are computed once, cached, and reused for all higher-layer interventions. It is important to note that this measurement was performed on RoBERTa-base (12 layers); the absolute time will be larger for larger models, but the relative reduction is expected to hold due to the

same caching principle. On RoBERTa-base, this overhead constitutes less than 2% of the total fine-tuning wall-clock time for a standard 10-epoch GLUE task run, making the profiling step a minor addition to the full adaptation pipeline.

5.4 Ablation Studies on Configuration Strategies

Having demonstrated the competitive performance of LoRA-Perturb on both encoder-only and decoder-only benchmarks, we now dissect the individual contributions of its two core components: (1) the rank allocation criterion and (2) the learning rate scheduling scheme. All experiments in this subsection are conducted on a 2000-sample subset of the SST-2 training set using $\text{RoBERTa}_{\text{base}}$, under the shared hyperparameter configuration detailed in Table 1. Note that the main GLUE results in Table 2 use the full training set; therefore, absolute accuracy values in this section are not directly comparable to those in Table 2, and the ablation results are intended to reveal relative trends across configuration strategies.

5.4.1 Rank Allocation Criteria

We compare three rank allocation strategies under a unified learning rate of 3×10^{-5} applied uniformly across all adapted layers:

- **Uniform Rank (Fixed LoRA):** All layers receive the baseline rank $r = 8$.
- **Gradient-based Allocation:** Ranks are distributed proportionally to the Fisher information trace estimated from a small subset of training data, following the approach of AdaLoRA [14].
- **Perturbation-based Allocation (Ours):** Ranks are distributed proportionally to the importance scores derived from our efficient perturbation framework.

Table 5: Comparison of rank allocation strategies on SST-2 accuracy (2000-sample subset) under a unified learning rate.

Strategy	**SST-2 Accuracy**
Uniform rank (Fixed LoRA)	0.915
Gradient-based allocation	0.922
Perturbation-based allocation	**0.928**

Settings: All experiments in this section use the shared hyperparameters in Table 1 on a 2000-sample subset of SST-2.

Table 5 reveals a clear hierarchy in allocation effectiveness. Uniform assignment yields the lowest accuracy (0.915), underscoring the fundamental inefficiency of treating all layers equally. Gradient-based allocation improves performance to 0.922, confirming that sensitivity-based heuristics provide a meaningful, albeit partial, signal for rank distribution. Our perturbation-based method further elevates accuracy to **0.928**, achieving higher accuracy than the gradient-based alternative by a meaningful margin. We attribute this gain to the perturbation-based nature of our measurement: whereas gradient norms capture correlational sensitivity, the perturbation provides a more direct measure of the *sensitivity* of a layer's adapted representation for producing correct predictions, offering a more principled foundation for architectural resource budgeting.

5.4.2 Effect of Differentiated Learning Rates

We next examine the impact of learning rate differentiation. For each rank allocation strategy, we contrast two learning rate policies:

- **Single Learning Rate (SLR):** All layers share the global learning rate $\eta = 3 \times 10^{-5}$.
- **Differentiated Learning Rate (DLR):** For each layer i, the learning rate is scaled inversely with its allocated rank: $\eta_i = \eta_{\text{base}} \cdot (r_{\text{base}}/r_i)$, where $r_{\text{base}} = 8$. This ensures that layers with higher rank—and thus larger effective update magnitude—receive proportionally smaller step sizes, promoting stable optimization dynamics.

Table 6: SST-2 accuracy (2000-sample subset) under combinations of rank allocation strategy and learning rate policy.

Strategy	**SLR (Unified)**	**DLR (Differentiated)**
Uniform rank	0.915	0.923
Gradient-based allocation	0.922	0.925
Perturbation-based allocation	0.928	**0.932**

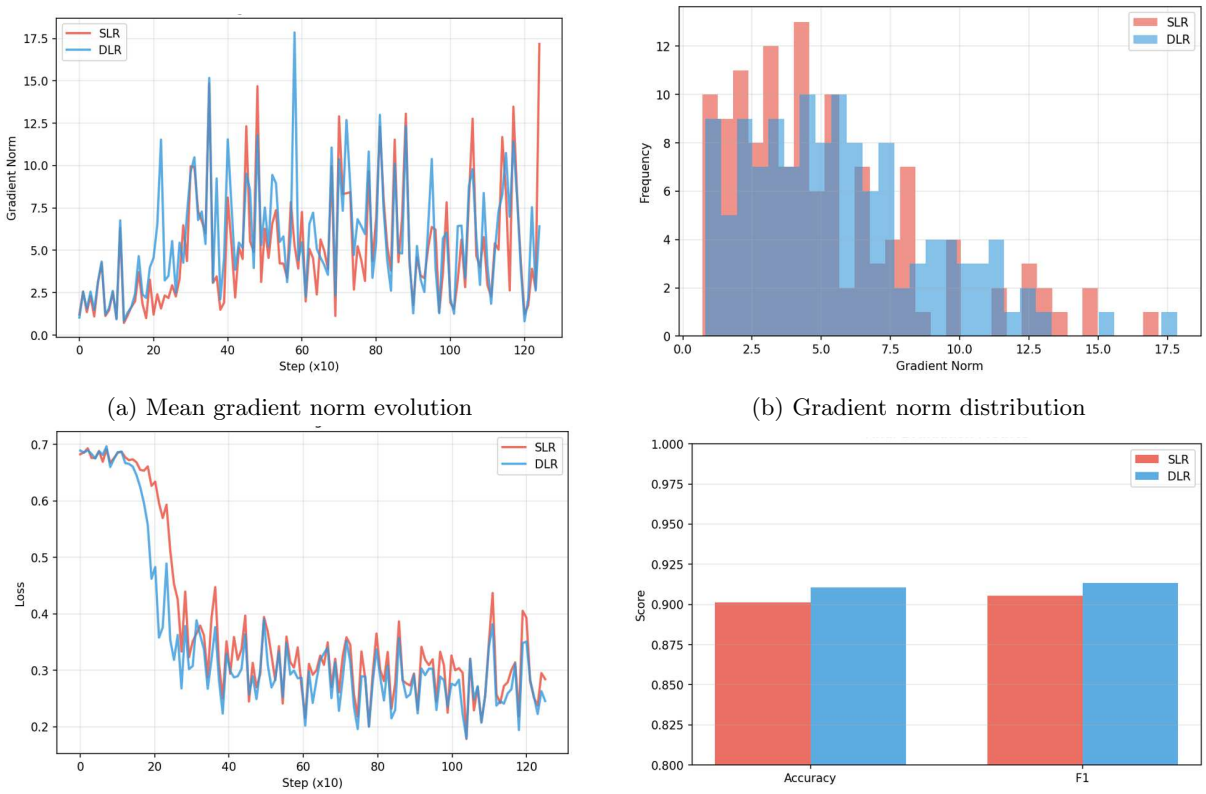
Several important observations emerge from Table 6:

1. **Consistent Benefit of DLR:** Across all three allocation strategies, transitioning from a unified to differentiated learning rate yields a consistent accuracy improvement (gains of +0.008, +0.003, and +0.004 for Uniform, Gradient-based, and Perturbation-based, respectively). This demonstrates that aligning optimization step sizes with layer-wise representational capacity is beneficial, independent of the specific rank assignment mechanism.
2. **Synergistic Effect with Perturbation-Based Allocation:** The highest absolute accuracy (**0.932**) is achieved by pairing perturbation-based rank allocation with differentiated learning rates. This combination jointly leverages the importance scores derived from perturbations—placing capacity precisely where it is most needed—and prudent optimization control that prevents unstable updates in high-capacity layers.
3. **Robustness of the Uniform Baseline:** Notably, even the uniform rank configuration benefits substantially from DLR (0.915 $\rightarrow$ 0.923). This suggests that a considerable portion of the instability inherent in LoRA fine-tuning can be mitigated purely through learning rate scaling, even in the absence of explicit rank differentiation.

In summary, the ablation results substantiate that both components of LoRA-Perturb—perturbation-informed rank allocation and differentiated learning rate scaling—are essential for realizing the full performance gains observed in our main experiments. The perturbation-based allocation provides a useful signal for *where* to allocate capacity, while the differentiated learning rate governs *how* that capacity should be optimized.

5.5 Training Dynamics: Gradient Norm Analysis

The ablation results in Section 5.4.2 established that differentiated learning rates (DLR) consistently improve final task performance when combined with our perturbation-based rank allocation. To elucidate the underlying optimization mechanisms responsible for this gain, we analyze the training dynamics of models fine-tuned with identical perturbation-based rank configurations but contrasting learning rate policies: a single global learning rate (SLR, $\eta = 3 \times 10^{-5}$ for all layers) and our proposed differentiated learning rate (DLR, $\eta_i \propto 1/r_i$). We monitor the gradient norms of the adapted LoRA matrices, the training loss trajectory, and validation performance throughout the fine-tuning process on the SST-2 task. All metrics are logged every 10 training steps over the first 125 steps (approximately one epoch),



(a) Mean gradient norm evolution

(b) Gradient norm distribution

(c) Training loss curve

(d) Final test accuracy

Figure 2: Training dynamics comparison between unified (SLR) and differentiated (DLR) learning rates under perturbation-based rank allocation. (a) Evolution of the mean gradient norm across all adapted LoRA layers over the first 125 training steps. (b) Histogram and kernel density estimate of gradient norm values aggregated over the monitored steps. (c) Training loss curve. (d) Final test accuracy on the SST-2 validation set. All experiments in this figure are conducted on 2000 samples from the SST-2 training set.

providing a high-resolution view of early-stage optimization behavior. Same configuration as in Section 5.4 is used.

Figure 2 presents a four-panel visualization of the optimization process. Panel (a) traces the mean gradient norm computed over all LoRA parameter matrices at each sampled step. The SLR curve exhibits more pronounced fluctuations than the DLR curve. While these fluctuations may reflect less stable updates, we do not have direct evidence linking them to inferior local minima. The DLR curve is smoother, which is consistent with the design intent of DLR: smaller step sizes for high-rank layers may prevent disproportionately large updates.

The stabilizing effect is further illustrated in Panel (b), which displays the frequency distribution of gradient norm values aggregated across all sampled steps and layers. Under SLR, the distribution is broader with a higher standard deviation ($\sigma = 3.44$) and a longer tail; under DLR, it is more concentrated ($\sigma = 3.33$). A tighter gradient norm distribution is often associated with more consistent parameter updates, but we cannot directly measure catastrophic forgetting or loss surface exploration from these plots alone.

Panel (c) shows that both learning rate schedules drive the training loss downward. The DLR trajectory descends more rapidly in the early phase and maintains a lower loss value than SLR. This observation is consistent with smoother optimization, but the causal link between this smoothness and final performance remains correlational.

Panel (d) shows a +0.92% absolute improvement in SST-2 accuracy for DLR over SLR under perturbation-based rank allocation. We note that this improvement, while consistent across repeated runs, is small, and we did not perform formal statistical testing across a large number of seeds; therefore, it should be interpreted as preliminary evidence of benefit rather than conclusive proof.

Collectively, the gradient norm analysis suggests that the perturbation-informed rank allocation identifies layers requiring additional capacity, and the inversely scaled learning rate helps control the update magnitudes in those layers. The resulting optimization process is characterized by smoother gradient flow in the monitored early phase, which may contribute to the observed performance gains.

6 Conclusion

In this paper, we addressed a fundamental tension in the landscape of parameter-efficient fine-tuning: while differentiated rank allocation has emerged as a promising strategy to overcome the uniformity constraint of standard LoRA, existing methods for quantifying layer importance typically rely on correlational proxies that may not reliably inform architectural decisions. Concurrently, perturbation-based intervention techniques from the interpretability literature offer a practical alternative, yet their direct application to large-scale fine-tuning has been impeded by prohibitive computational overhead.

To bridge this gap, we introduced LoRA-Perturb, an efficient perturbation-based framework for layer importance estimation that enables differentiated LoRA configuration. Our approach makes three core contributions. First, we proposed an efficient perturbation-based protocol that quantifies the importance of each transformer layer by measuring the drop in prediction confidence following minimal noise injection into its LoRA matrices. Second, we developed a three-fold optimization strategy—comprising random subset downsampling, single-pass activation caching, and localized forward reuse—that reduces measurement FLOPs by approximately 50% and wall-clock time by about 38% on a 12-layer encoder. Third, we leveraged the resulting layer-wise importance scores to warm-start fine-tuning with a differentiated configuration: layers with higher perturbation sensitivity receive higher ranks paired with lower learning rates, while less sensitive layers are assigned lower ranks and correspondingly higher learning rates.

Through experiments spanning encoder-only ($\text{RoBERTa}_{\text{base}}$) and decoder-only (Llama-2-7B) architectures, we demonstrated that LoRA-Perturb improves over standard uniform LoRA on average and

achieves competitive performance among LoRA variants. On GLUE, it attained the highest average score (89.4) among compared methods, though it was not best on every task. On both MT-Bench and GSM8K, LoRA-Perturb achieved the highest scores among the compared LoRA variants, slightly above DoRA and LoRA+, though the differences are within the reported standard deviations and should be interpreted with caution. Ablation studies confirmed that perturbation-informed rank allocation and differentiated learning rates contribute both independently and synergistically to these gains. Furthermore, analyses of gradient norm dynamics revealed that differentiated learning rates lead to smoother optimization trajectories.

The implications of this work extend beyond the specific instantiation of LoRA. Our findings reinforce the broader principle that perturbation-based layer importance can productively inform architectural and optimization decisions in deep learning, provided that the measurement apparatus is engineered for efficiency. The demonstrated ability to recover a large fraction of full fine-tuning performance on complex reasoning tasks with a minuscule parameter budget, while maintaining competitive conversational quality, suggests that importance-aware resource allocation is particularly valuable in resource-constrained or multi-task deployment scenarios.

Several directions merit future investigation. First, extending the perturbation measurement framework to other PEFT paradigms beyond LoRA—such as adapter-based methods or prompt tuning—could reveal whether the same layer importance patterns generalize across different adaptation mechanisms. Second, exploring dynamic reallocation of ranks during the course of training, rather than relying on a static warm-start configuration, may further optimize the trade-off between capacity and efficiency. Third, investigating the interaction between our perturbation-based allocation strategy and quantization techniques could unlock additional memory and latency reductions for on-device deployment. Finally, a more systematic examination of importance patterns across diverse model families and scales may yield architectural insights applicable to the design of future pre-trained models themselves.

In summary, LoRA-Perturb provides a practical solution for differentiated LoRA fine-tuning, demonstrating that perturbation-based layer importance can be measured efficiently and leveraged effectively to improve both final performance and training stability. While it does not uniformly surpass all existing variants on every metric, it offers consistent gains over standard LoRA and establishes a strong baseline for importance-driven parameter allocation. We hope this work encourages further integration of perturbation-based analysis techniques into the design of efficient adaptation methods for large language models.

Funding

This work was supported by the Wenzhou Municipal Science and Technology Bureau under Grant No. GC20250003.

Algorithm Pseudocode

Algorithm 1 LoRA-Perturb: Efficient Perturbation-Based Layer Importance for Differentiated LoRA

Require: Pre-trained model $\mathcal{M}$ with L layers, training set $\mathcal{D}_{\text{train}}$, rank budget R_{total}, sample size N, warm-up steps T, noise parameters σ, p_{mask}.
Ensure: Differentiated LoRA configuration $\mathcal{C} = \{(r_l, \eta_l)\}_{l=1}^{L}$.
Select a random subset $\mathcal{D}_s \subset \mathcal{D}_{\text{train}}$ of size N (e.g., 100–500 samples).
// Stage 1: Task-Adaptation Warm-up
Initialize LoRA $\{A_l, B_l\}_{l=1}^{L}$ with uniform rank $r_{\text{init}} = 8$.
for $t = 1$ to T **do**
    Update LoRA parameters via gradient descent on mini-batches from $\mathcal{D}_s$.
end for
Obtain warm-up model $\mathcal{M}_{\text{warm}}$.
// Stage 2: Low-Overhead Perturbation Measurement
for each $x \in \mathcal{D}_s$ **do**
    Full forward pass through $\mathcal{M}_{\text{warm}}$; cache $\{h_{l-1}^{\text{cache}}\}_{l=1}^{L}$ and record $P_{\mathcal{M}_{\text{warm}}}(y \mid x)$.
end for
for each layer $l = 1$ to L **do**
    $s_l \leftarrow 0$.
    for each $x \in \mathcal{D}_s$ **do**
        Localized forward from layer l using cached h_{l-1}^{cache}; inject noise:
        $\tilde{h}_l^{\text{warm}} = W_{0,l} h_{l-1} + \Delta W_l^{\text{warm}} h_{l-1} + \epsilon \cdot \mathcal{N}(0, \sigma^2 I)$.
        Continue forward through layers $l+1$ to L; obtain $P_{\mathcal{M}_{\text{warm}}}(y \mid \text{do}(h_l))$.
        $s_l \leftarrow s_l + \left[P_{\mathcal{M}_{\text{warm}}}(y \mid x) - P_{\mathcal{M}_{\text{warm}}}(y \mid \text{do}(h_l))\right]$.
    end for
    $s_l \leftarrow s_l / N$.
end for
Apply rectification: $s_l \leftarrow \max(0, s_l)$ for all l; if sum is zero, set $s_l = 1/L$.
Normalize $\bar{s}_l = s_l / \sum_{j=1}^{L} s_j$.
// Stage 3: Differentiated Full Fine-tuning Configuration
for each layer $l = 1$ to L **do**
    $r_l^{\text{raw}} \leftarrow r_{\min} + (r_{\max} - r_{\min}) \cdot \bar{s}_l$, with $r_{\min} = 4$, $r_{\max} = 16$.
    Round r_l^{raw} to nearest integer and adjust to match R_{total} via largest fractional remainders.
    $\eta_l \leftarrow \eta_{\text{base}} \cdot \frac{r_{\text{base}}}{r_l}$, where $r_{\text{base}} = 8$.
end for
return $\mathcal{C} = \{(r_l, \eta_l)\}_{l=1}^{L}$.

Data availability

The datasets analyzed during the current study are publicly available. The GLUE benchmark datasets (MRPC, CoLA, STS-B, SST-2, QNLI) are available from the GLUE benchmark repository. MT-Bench and GSM8K datasets are available from their respective public repositories. The code and scripts used for the experiments are available from the corresponding author upon reasonable request.

Author contributions

H.J. and Z.Z. conceived the study. H.J. developed the methodology and wrote the main manuscript text. Z.Z., Y.G., and L.T. conducted the experiments and data analysis. X.L. and Z.L. contributed to the experimental design and results interpretation. Z.L. supervised the project and acquired funding. All authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Figure Legends

Figure 1. Comparison of different LoRA initialization and rank allocation strategies. **(a) LoRA:** Fixed rank for all layers, $A_0 \sim \mathcal{N}(0, \sigma^2)$, $B_0 = 0$. **(b) AdaLoRA:** Dynamically prunes relatively unimportant ranks based on importance scores during training, with the same initialization as (a). **(c) DoRA:** Decomposes pre-trained weight into magnitude and direction components, allocates ranks to the direction component, with $A_0 \sim \mathcal{N}(0, \sigma^2)$, $B_0 = 0$. **(d) LoRA-Perturb (Ours):** Allocates ranks based on probability variation of perturbation.

Figure 2. Training dynamics comparison between unified (SLR) and differentiated (DLR) learning rates under perturbation-based rank allocation. **(a)** Evolution of the mean gradient norm across all adapted LoRA layers over the first 125 training steps. **(b)** Histogram and kernel density estimate of gradient norm values aggregated over the monitored steps. **(c)** Training loss curve. **(d)** Final test accuracy on the SST-2 validation set. All experiments in this figure are conducted on 2000 samples from the SST-2 training set.

References

[1] 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, 2021.

[2] Justin Zhao, Timothy Wang, Wael Abid, Geoffrey Angus, Arnav Garg, Jeffery Kinnison, Alex Sherstinsky, Piero Molino, Travis Addair, and Devvret Rishi. Lora land: 310 fine-tuned llms that rival gpt-4, a technical report, 2024.

[3] Juneyoung Park, Eunbeen Yoon, and Seongwan Kim. Jaeho Lee. Lcsb: Layer-cyclic selective backpropagation for memory-efficient on-device llm fine-tuning, 2026.

[4] Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. Lora vs full fine-tuning: An illusion of equivalence, 2025.

[5] Xinyi Wang, Lirong Gao, Haobo Wang, Yiming Zhang, and Junbo Zhao. Floe: Fisher-based layer selection for efficient sparse adaptation of low-rank experts, 2025.

[6] Zequan Liu, Jiawen Lyn, Wei Zhu, Xing Tian, and Yvette Graham. Alora: Allocating low-rank adaptation for fine-tuning large language models, 2024.

[7] Wenjun Ke, Jiahao Wang, Peng Wang, Jiajun Liu, Dong Nie, Guozheng Li, and Yining Li. Unveiling lora intrinsic ranks via salience analysis. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 131575–131595. Curran Associates, Inc., 2024. doi: 10.52202/079017-4182.

[8] Zikai Zhang, Ping Liu, Jiahao Xu, and Rui Hu. Fed-hello: Efficient federated foundation model fine-tuning with heterogeneous lora allocation, 2025.

[9] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc., 2020.

[10] Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan,

editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9574–9586. Curran Associates, Inc., 2021.

[11] Yara Shamshoum, Nitzan Hodos, Yuval Sieradzki, and Assaf Schuster. Compact: Compressed activations for memory-efficient llm training, 2025.

[12] Antoine Simoulin, Namyong Park, Xiaoyi Liu, and Grey Yang. Memory-efficient fine-tuning of transformers via token selection, 2025.

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

[14] 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, 2023.

[15] 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, 2019.

[16] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

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

[18] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 17783–17806. PMLR, 21–27 Jul 2024.

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

[20] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.