Sensitivity-LoRA : Low-Load Sensitivity-Based Fine-Tuning for Large Language Models

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have revolutionized various domains, including everyday applications and scientific research. However, adapting LLMs from general-purpose models to specialized tasks remains challenging, particularly in resource-constrained environments. Parameter-Efficient Fine-Tuning (PEFT), especially Low-Rank Adaptation (LoRA), has emerged as a promising approach to fine-tuning LLMs while reducing computational and memory overhead. Despite its advantages, existing LoRA techniques still struggle with computational inefficiency, complexity, and instability, limiting their practical applicability. To address these limitations, we propose Sensitivity-LoRA, an efficient fine-tuning method that dynamically allocates ranks to weight matrices based on both their global and local sensitivities. It leverages the second-order derivatives (Hessian Matrix) of the loss function to effectively capture weight sensitivity, enabling optimal rank allocation with minimal computational overhead. Our experimental results have demonstrated robust effectiveness, efficiency and stability of Sensitivity-LoRA across diverse tasks and benchmarks.

1 Introduction

Large language models (LLMs) have emerged as revolutionary tools, delivering state-of-the-art (SOTA) performance across a wide spectrum of tasks and applications (Ding et al., 2022; Qin et al., 2023; Zhu et al., 2023b,a; Li et al., 2023; Zhang et al., 2023a; Huang et al., 2023; Wang et al., 2023). Despite these advancements, fine-tuning remains a critical technique for effectively adapting LLMs from general-purpose models to specialized applications, especially in resource-constrained environments. However, full-parameter fine-tuning can be prohibitively resource-intensive, requiring significant computational power and GPU capacity. To address this limitation, the research community introduced parameter-efficient fine-tuning (PEFT) (Houlsby et al., 2019; Lester et al., 2021; Li and Liang, 2021; Zaken et al., 2022; Hu et al., 2022a), which aims to balance accuracy and efficiency by selectively updating a subset of model parameters. A decent amount of PEFT methods have been validated to be effective across a variety of models and tasks, often achieving results comparable to those of full-parameter fine-tuning (Lester et al., 2021; Zaken et al., 2022; Hu et al., 2022a).

Among these PEFT methods, the Low-Rank Adaptation (LoRA) method based on reparameterization (Hu et al., 2022b) is considered one of the most efficient and effective approaches. It utilizes low-rank decomposition to approximate the updates of model weights. This strategy draws from prior research demonstrating that the learned over-parametrized models in fact reside on a low intrinsic dimension (Li et al., 2018; Aghajanyan et al., 2020). During the training process, the update of the weight matrix (ΔW), can be approximated as the product of two smaller matrices B and A, that is:

$$\Delta W \approx B \cdot A \tag{1}$$

where $\Delta W \in R^{d_1 \times d_2}$, $A \in R^{r \times d_2}$ and $B \in R^{d_1 \times r}$ with $r \ll \{d_1, d_2\}$. Through this method, it can approximate the update of the weight matrix with fewer parameters, thereby improving the parameter efficiency of the model. However, the full potential of LoRA remains constrained by its inherent design limitations. Specifically, it assumes a uniform rank *r* for each incremental matrix, not accounting for the varying significance of weight matrices across different modules and layers (Hu et al., 2023; Zhang et al., 2023b).

To address this limitation, dynamic rank allocation has emerged as a key solution by allocating the rank r to each different module or layer according to its specific requirements. Existing methods achieve this through three main approaches:

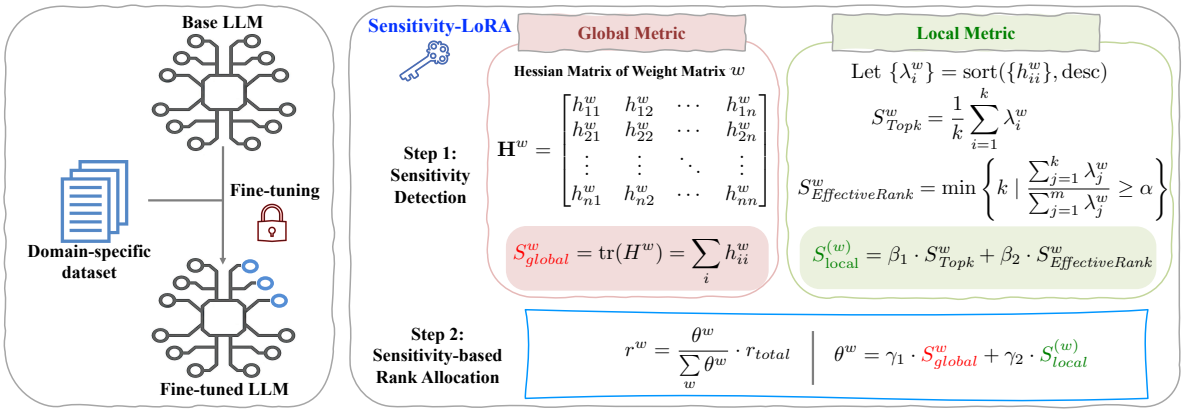


Figure 1: Pipeline of the Sensitivity-LoRA Method: **Step 1 - Sensitivity detection** via Hessian-based metrics, including global and local sensitivity measures. ($h_{ij}^w = \frac{\partial^2 E^w}{\partial w_i \partial w_j}$, $h_{ii} = \frac{\partial^2 E^w}{\partial w_i^2}$, where E^w denotes the change in loss function regarding weight matrix w; $\mathrm{tr}(H^w)$ denotes the trace of H^w.) **Step 2 - Dynamic rank allocation** based on global and local sensitivity. (r^w denotes the allocated rank of weight matrix w, r_{total} denotes the total rank of all weight matrices in the model.)

singular value decomposition (SVD), single-rank decomposition (SRD), and rank sampling. SVD-based methods (Zhang et al., 2023c; Hu et al., 2023; Zhang et al., 2023b) decompose matrix BA into an SVD form and selectively truncate the singular values in order to allocate the matrix rank. However, this process is computationally expensive and requires additional memory to store singular values and vectors. SRD-based methods (Mao et al., 2024; Zhang et al., 2024; Liu et al., 2024) decompose matrix BA into single-rank components and allocate the ranks by selecting the proper components. However, optimizing single-rank components and the pruning process can increase computational complexity, potentially offsetting efficiency gains. Rank sampling-based methods (Valipour et al., 2022) allocate ranks directly by random sampling. However, the randomness introduced by sampling could increase potential instability in the training.

In view of these challenges and opportunities, we propose Sensitivity-LoRA, which can rapidly allocate rank to the weight matrix based on the sensitivity of the parameters, without incurring a significant computational load. Specifically, we utilize the second derivatives of the loss function with respect to the parameters (Hessian matrix) to ascertain the sensitivity of each parameter within the weight matrix. To comprehensively evaluate the sensitivity of the parameter matrix, we employ metrics such as the trace of the Hessian matrix, *Topk* and *Effective Rank* to measure its global and local sensitivities respectively. By integrating various metrics, we determine the rank allocation weights corresponding to the weight matrices to achieve rank allocation. The efficiency, stability, and generality of our approach have been validated through extensive experiments on various tasks, such as sentiment analysis, natural language inference, question answering, and text generation.

In summary, the main contributions of our paper are listed as follows:

- We introduce the second derivatives of the loss function with respect to the weight matrix to measure their sensitivity.
- We achieve rank allocation by taking into account both the global and local sensitivity of the weight matrix.
- Extensive experiments demonstrate the effectiveness, stability, and efficiency of our method.

2 Related Work

Existing PEFT approaches can be classified into four main types in terms of memory efficiency, storage efficiency, and inference overhead, as follows:

2.1 Additive PEFT

Additive PEFT introduces lightweight modules into the model architecture, such as adapters and soft prompts, while keeping the pre-trained backbone frozen. Adapters add small networks with down-projection and up-projection matrices, enabling task-specific learning with minimal parameter updates (Houlsby et al., 2019; Lester et al., 2021).

Soft prompts prepend learnable embeddings to the input sequence, allowing fine-tuning by modifying input activations only (Li and Liang, 2021; Zaken et al., 2022). These methods typically require updating less than 1% of the total parameters, significantly reducing computation and memory costs, making them ideal for resource-constrained environments (Hu et al., 2022a).

2.2 Selective PEFT

Selective PEFT fine-tunes a subset of the existing parameters in a pre-trained model, rather than adding new modules. It employs binary masks to identify and update only the most important parameters while keeping the majority frozen. Techniques like Diff pruning and FishMask leverage Fisher information or parameter sensitivity analysis to select critical parameters for fine-tuning (Zaken et al., 2022; Li and Liang, 2021). This approach avoids increasing model complexity and is particularly suited for scenarios where only a small fraction of the model contributes significantly to performance.

2.3 Reparameterized PEFT

Reparameterized PEFT utilizes low-rank parameterization techniques to represent model weights in a reduced form during training. LoRA (Low-Rank Adaptation) is a prominent example, introducing low-rank matrices to fine-tune specific weights while maintaining high inference efficiency (Hu et al., 2022a). Other methods, such as Compacter, use the Kronecker product for parameter reparameterization, further reducing memory requirements and computational costs (Houlsby et al., 2019). Reparameterized PEFT is highly effective for large-scale models where resource constraints are critical.

2.4 Hybrid PEFT

Hybrid PEFT combines the strengths of Additive, Selective, and Reparameterized PEFT methods into a unified framework. For example, UniPELT integrates LoRA, adapters, and soft prompts, allowing dynamic selection of the most suitable module for specific tasks through gating mechanisms (Zaken et al., 2022). This hybrid approach enhances adaptability and task performance by leveraging the complementary advantages of different PEFT strategies (Li and Liang, 2021; Hu et al., 2022a).

3 Methodology

In this section, we firstly introduce the concept of weight sensitivity, followed by a formal definition of global and local sensitivity metrics of weight matrices. Next, we propose effective allocation strategies to optimize the dynamic rank allocation process based on these sensitivity metrics. The pipeline of our method is presented in figure 1.

3.1 Weight Sensitivity

Consider a neural network whose dynamics is driven by a collection of parameters w and a loss function E, which guides its learning dynamics. When a small perturbation δw is introduced to the parameters, the resulting change in the loss function can be expressed using a Taylor series expansion up to the second-order term, with higher-order terms captured by $O(\|\delta w\|^3)$ as follows:

$$E(w+\delta w) = E(w)+g^T\delta w+\frac{1}{2}\delta w^T H\delta w+O(\|\delta w\|^3) \tag{2}$$

where g denotes the gradient vector of the loss function E with respect to the parameters w, indicating the rate of change of the loss function in the direction of each parameter. H represents the Hessian matrix of the loss function E, which is a matrix of second-order partial derivatives and contains information about the curvature of the loss function at the current parameter point. The change in the loss function ΔE can be represented by the following expression:

$$\Delta E = g^\top \delta \mathbf{w} + \frac{1}{2}\delta \mathbf{w}^\top H \delta \mathbf{w} + O(\|\delta_w\|^3) \tag{3}$$

By expanding the components of ΔE, we have:

$$\Delta E = \sum_i g_i \delta w_i + \frac{1}{2}\sum_{i,j} h_{ij}\delta w_i \delta w_j + O(\|\delta_w\|^3) \tag{4}$$

where g_i and h_{ij} are the gradient and Hessian elements, respectively.

For a well-trained neural network, when the parameter w is located at a local minimum of the loss function, the gradient g becomes zero. Then, the above equation can be simplified to

$$\Delta E = \frac{1}{2}\sum_{i,j} h_{ij}\delta w_i \delta w_j + O(\|\delta_w\|^3) \tag{5}$$

Additionally, the parameters w are nearly independent at the minimum, the Hessian matrix H tends to be diagonal-dominant, meaning that the

off-diagonal elements, which represent interactions between different parameters, become negligible. Then, the above equation can be simplified to

$$\Delta E \approx \frac{1}{2} \sum_i h_{ii} \delta_{w_i}^2 + O(\|\delta_w\|^3) \tag{6}$$

Given that the perturbation in the weights ($\delta \mathbf{w}$) is sufficiently small, the higher-order term becomes negligible compared to the quadratic term, and therefore can be disregarded. Consequently, the above formula can be further simplified to

$$\Delta E \approx \frac{1}{2} \sum_i h_{ii} \delta_{w_i}^2 \tag{7}$$

Consequently, the diagonal elements of the Hessian matrix serve as a reliable indicator of weight sensitivity. Specifically, the diagonal elements of the Hessian matrix represent the local curvature of the loss function in the direction of each parameter. Larger diagonal elements indicate more dramatic changes in the loss function in that parameter direction, thereby showing that the parameter has a more significant impact on model performance. Furthermore, the methods for approximating the Hessian matrix have become increasingly sophisticated, ensuring that they do not place a substantial load on computational resources.

3.2 Rank Allocation Metric

3.2.1 Global Metric

The global sensitivity measurement aims to evaluate the overall impact of an entire parameter (or weight) matrix on model output. It quantifies how variations in this weight matrix affect the loss function. To capture this dynamics, the Hessian matrix, which consists of the second-order partial derivatives of the loss function with respect to the weight matrix, is used. Given that the Hessian matrix tends to be diagonal-dominant at the minimum, its trace can serve as an effective global sensitivity indicator. Formally, the global sensitivity S_{global}^w of weight matrix w can be defined as:

$$S_{global}^w = \text{tr}(H^w) = \sum_i h_{ii}^w \tag{8}$$

where h_{ii}^w is the i-th diagonal element of the Hessian matrix H^w, and $\text{tr}(H^w)$ denotes the trace of H^w. Since the diagonal elements reflect the impact of individual parameter changes on the loss function, a larger trace value indicates that the model is more sensitive to its changes. This suggests more parameters make significant contributions to the changes in the loss function, emphasizing their role in model performance.

3.2.2 Local Metric

While certain weight matrices might have low overall sensitivities, specific weight elements within these matrices can still have high sensitivity, significantly impacting model performance. As such, it is essential to account for local sensitivity to capture finer-grained variations in parameter influence on the loss function. To address this, we introduce two metrics: *Topk* and *Efficient Rank*.

The *Topk* metric approximates local sensitivity of a weight matrix by averaging its largest *k* diagonal elements of Hessian matrix, based on the assumption that most of the matrix's energy or sensitivity is concentrated in these large values. By focusing on *Top k* diagonal elements, the *Topk* metric can guide us to prioritize these critical weights during weight pruning or optimization processes. It reduces computational complexity while preserving the most impactful weights for model performance. The computation formula for the *Topk* metric of weight matrix w is as follows:

$$S_{Topk}^w = \frac{1}{k} \sum_{i=1}^{k} \lambda_i^w \tag{9}$$

where λ_i^w represents the diagonal elements of Hessian matrix H^w sorted in descending order, and k denotes the number of diagonal elements selected.

The *Effective Rank* metric determines the minimum rank that captures most of the energy of a weight matrix based on the cumulative contribution of the diagonal elements of its Hessian matrix. By establishing a threshold for the cumulative contribution rate (such as 0.9 or 0.95), the *Effective Rank* metric identifies the minimum number of eigenvalues needed to achieve this threshold, thereby appropriately ranking the weight matrix. The key benefit of this metric is ensuring the stability of the rank allocation process. The formula for *Effective Rank* of weight matrix w is as follows:

$$S_{EffectiveRank}^w = \min \left\{ k \mid \frac{\sum_{j=1}^{k} \lambda_j^w}{\sum_{j=1}^{m} \lambda_j^w} \geq \alpha \right\} \tag{10}$$

where λ_j^w is the j-th diagonal element of H^w in non-increasing order, m is the total number of diagonal elements, and k is the minimum number of diagonal elements required for the cumulative contribution rate to reach the threshold α.

Method	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
LoRA	87.26	93.46	87.01	58.83	92.95	90.50	79.42	91.03	85.06
S_g-LoRA	87.15	94.61	88.48	60.83	93.08	90.54	79.78	91.03	85.69
S_T-LoRA	86.94	93.81	88.48	60.82	92.75	90.61	79.06	91.20	85.46
S_E-LoRA	87.54	94.04	89.46	63.07	92.77	90.58	80.51	91.16	86.14
Ours	**87.84**	**95.11**	**89.53**	**63.21**	**93.08**	**90.72**	**80.64**	**91.31**	**86.43**

Table 1: The comparative results of Sensitivity-LoRA and baseline methods for fine-tuning RoBERTa-base on the eight datasets of the GLUE benchmark. The bolded values represent the optimal outcomes. We reported the overall accuracy for MNLI (both matched and mismatched), the Matthew's correlation for CoLA, the Combine Score for STS-B, and the accuracy for the other tasks. (S_g-LoRA, S_T-LoRA and S_E-LoRA represent LoRA fine-tuning using the global metric, *Topk* metric, and *Efficient Rank* metric for rank allocation, respectively.)

To ensure the effectiveness and stability, we integrate *Topk* and *Efficient Rank* metrics together to define the local sensitivity metric S^w_{local} of weight matrix w as follows:

$$S^w_{local} = \beta_1 \cdot S^w_{Topk} + \beta_2 \cdot S^w_{EffectiveRank} \quad (11)$$

where $\beta_1 + \beta_2 = 1$.

3.3 Rank Allocation Strategy

Taking into account both global and local metrics, we define a refined rank allocation strategy to determine the rank allocation weights θ^w of weight matrix w by integrating global and local sensitivities:

$$\theta^w = \gamma_1 \cdot S^w_{global} + \gamma_2 \cdot S^w_{local} \quad (12)$$

subject to the constraints $\gamma_1 + \gamma_2 = 1$. Hence, we can derive the formula for rank allocation as follows:

$$r^w = \frac{\theta^w}{\sum_w \theta^w} \cdot r_{total} \quad (13)$$

where r^w denotes the rank allocated to weight matrix w, and r_{total} represents the total rank of all weight matrices in the model.

4 Experiments

4.1 Experimental Setup

Benchmarks We evaluate the performance of Sensitivity-LoRA across diverse tasks, including NLG (Natural Language Generation) and NLU (Natural Language Understanding). Specifically, for the NLU tasks, we select RoBERTa-base (Liu, 2019) as the base model and evaluate its performance on eight sub-tasks of the GLUE (General Language Understanding Evaluation) benchmark (Wang, 2018): MNLI (Williams et al., 2017), SST-2 (Socher et al., 2013), MRPC (Dolan and Brockett, 2005), CoLA (Warstadt, 2019), QNLI (Rajpurkar et al., 2018), QQP [1], RTE (Wang, 2018), and STS-B (Cer et al., 2017). For the NLG tasks, we assess the performance of the proposed method with GPT-2 Large (Radford et al., 2019) on E2E benchmark (Novikova et al., 2017).

Baselines The LoRA (Hu et al., 2022b) with a uniform rank allocation is adopted as the baseline for comparison with the proposed Sensitivity-LoRA. We maintain the same hyperparameter settings as those used in LoRA, varying only the rank allocation method. We further compare the fine-tuning methods that allocate ranks based solely on global or local sensitivity metrics with the proposed Sensitivity-LoRA to highlight its effectiveness. Specifically, S_g-LoRA, S_T-LoRA and S_E-LoRA represent LoRA fine-tuning methods that allocate ranks using the global metric, *Topk* metric, and *Efficient Rank*, respectively.

Implementation Details Our code is implemented using the PyTorch (Paszke et al., 2019) framework and Transformers (Wolf, 2020) libraries, with all experiments conducted on four NVIDIA A100 GPUs. We fine-tune GPT-2 Large (36 layers) and RoBERTa-base (12 layers) using Sensitivity-LoRA. For a fair comparation, we adhere to the same total rank r_{total} as specified in LoRA. Specifically, for GPT-2 Large, we set $r_{total} = 144$, while for RoBERTa-base, we set $r_{total} = 96$. We designate the local metric S_{Topk} with k set to half of the total number of diagonal elements, and set the parameter α in the *Efficient Rank* metric to 0.8. Considering the scale characteristics of the different metrics, we set the parameter β_1 to 0.45, β_2 to 0.55, γ_1 to 0.005, and γ_2 to 0.095.

[1] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

Method	BLEU	NIST	MET	ROUGE-L	CIDEr
LoRA	67.15	8.58	46.27	69.02	2.39
S_g-LoRA	67.61	8.61	46.24	69.34	2.38
S_T-LoRA	67.37	8.60	46.14	69.47	2.41
S_E-LoRA	67.94	8.64	46.14	69.40	2.40
Ours	**67.99**	**8.65**	**46.58**	**69.64**	**2.41**

Table 2: The results of GPT-2 Large on the E2E dataset for the baseline and Proposed method. For all metrics, higher values indicate better performance, and the bold values represent the best results. (S_g-LoRA, S_T-LoRA and S_E-LoRA represent LoRA fine-tuning using the global metric, *Topk* metric, and *Efficient Rank* metric for rank allocation, respectively.)

4.2 Main Result

At first, we assess the performance of Sensitivity-LoRA on NLU tasks by fine-tuning the RoBERTa-base model on the eight tasks of the GLUE benchmark. The overall accuracy for MNLI, the Matthew's correlation for CoLA, the Combined Score for STS-B, and the accuracy for the other tasks are reported in table 1. Sensitivity-LoRA demonstrates outstanding performance in a variety of natural language understanding tasks. Compared to baseline methods, our approach achieve an average score of 86.43 over the eight datasets, representing a gain of 1.37 over the LoRA. The evaluation metrics of each dataset also significantly surpass those of other methods. Sensitivity-LoRA leverages the second-order derivatives of the loss function to extract weight-wise importance metrics, incorporating both local and global sensitivity. Based on these metrics, it dynamically determines the optimal rank allocation, thereby achieving exceptional performance.

Next, we fine-tune the GPT-2 Large model on the E2E (Novikova et al., 2017) datasets for NLG task evaluation. The E2E dataset is commonly employed for end-to-end data-to-text generation tasks. We train the model on 42,000 data-to-text pairs from the restaurant domain. To comprehensively evaluate both the baseline methods and Sensitivity-LoRA, we utilized five evaluation metrics: BLEU (Papineni et al., 2002), NIST (Lin and Och, 2004), METEOR (Banerjee and Lavie, 2005), ROUGE-L (Lin, 2004), and CIDEr (Vedantam et al., 2015). Table 2 presents the experimental results of Sensitivity-LoRA on the E2E dataset. We evaluate the impact of global and local metrics for rank allocation, alongside a combined rank allocation strategy incorporating both types of information. The experimental results indicate that our proposed rank allocation strategy, achieve superior performance across all evaluation metrics. Although relying solely on global or local information for rank allocation can improve model performance to some extent, its performance on several metrics (METEOR and CIDEr) still falls slightly short of the LoRA model. This may be because an overemphasis on either global or local information diminishes the importance of certain critical weights, thereby affecting the overall performance of the model.

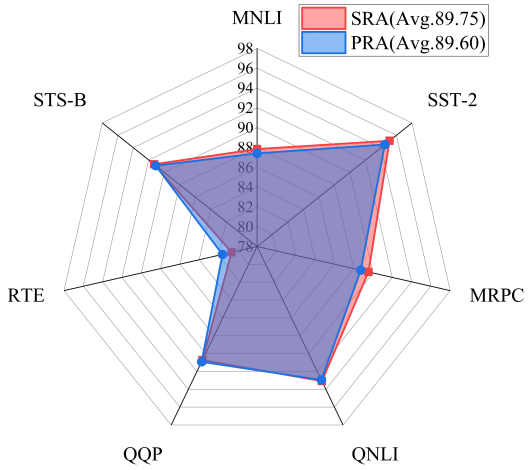


Figure 2: Comparison of the evaluation metrics for RoBERTa-base fine-tuned on seven datasets from the GLUE benchmark, using both PRA and SRA rank assignment methods.

4.3 Comparison of Rank Assignment Methods

In this section, we compare two rank allocation methods for model weights, based on global or local sensitivity metrics. The Progressive Rank Assignment (PRA) method first sorts the global or local sensitivity metrics in descending order, subsequently allocating ranks progressively within a specified range. Weights with higher sensitivity are assigned higher ranks, with rank decrementing in 20% intervals. The rank ranges for GPT-2

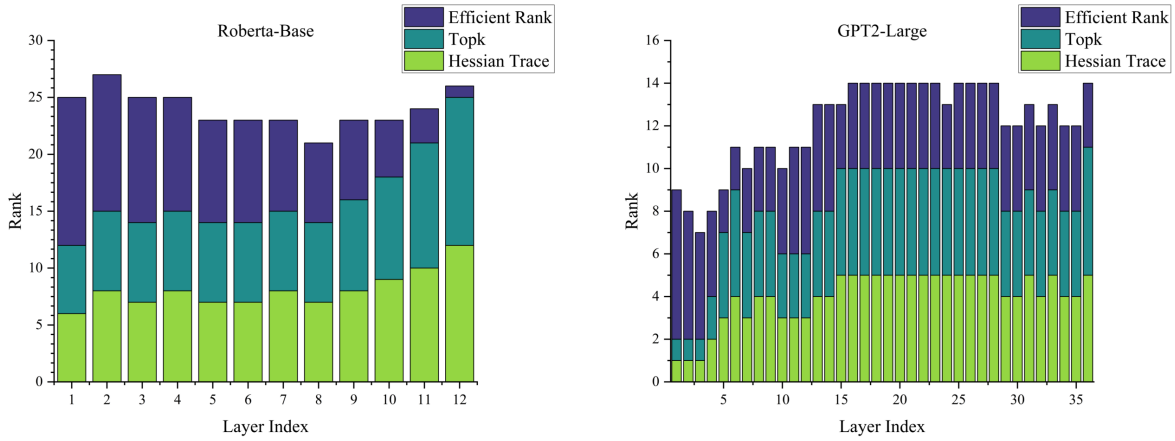


Figure 3: The rank allocation for each layer of GPT-2 Large and RoBERTa-base under different rank allocation strategies. Different colors represent different allocation strategies, and the height of each bar in the histogram corresponds to the rank allocated to that layer by the respective strategy.

Large and RoBERTa-base are 2~6 and 6~10, respectively. The Scaled Rank Assignment (SRA) method (mentioned in section 3.3) allocates ranks according to the proportion of each weight's global or local sensitivity relative to the model's total sensitivity. To visually compare the effectiveness of these two allocation methods, we apply both strategies to the global and local sensitivity metrics and subsequently combine them using the corresponding rank allocation strategy. We then fine-tune RoBERTa-base on seven datasets from the GLUE benchmark. As depicted in figure 2, the evaluation metrics for both allocation methods are compared across datasets. SRA and PRA achieve average scores of 89.75 and 89.60, respectively. Although SRA perform marginally worse than PRA on the RTE dataset, it surpass PRA on all other datasets. This highlights the effectiveness of the SRA allocation strategy. Consequently, we adapt the SRA method for rank allocation in this paper.

4.4 Optimized Rank Allocation

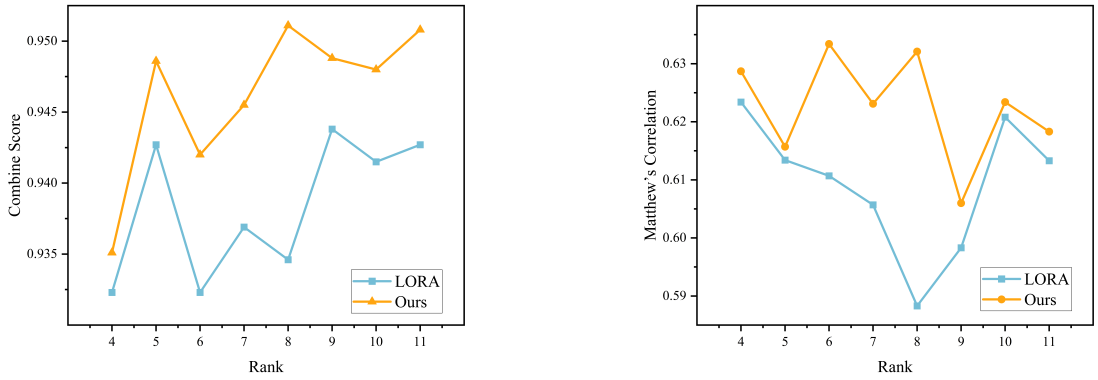
As shown in figure 3, we visualize the global and local rank allocation results for GPT-2 Large and RoBERTa-base under the SRA rank assignment method described in section 3.3. From the figure 3, it is clear that the global sensitivity metric, Hessian Trace, allocates a larger rank budget to the intermediate and deeper layers of the model, with less emphasis on the initial layers. The local sensitivity metric, *Topk*, primarily allocates rank to the middle layers of the model. In contrast, the *Efficient Rank* approach assigns higher ranks to the initial layers, while exhibiting a decreasing trend in rank allocation for subsequent layers. Each of these three sensitivity metrics emphasizes different aspects, indicating that relying on a single source of information for decision-making is insufficient. This underscores the necessity of Sensitivity-LoRA in integrating these diverse information sources for dynamic rank allocation.

4.5 Analysis of Different Rank Budgets

We further analyze the performance of Sensitivity-LoRA and the baseline method, LoRA, under various rank budget scenarios. Figure 4 illustrates the changes in evaluation metrics for RoBERTa-base on the SST-2 and CoLA datasets for different rank budgets. We evaluate the model's performance under eight rank budgets, ranging from 4 to 11. For Sensitivity-LoRA, the ranks in the figure 4 represent the average rank of each model layer. As seen in the figure 4, the proposed Sensitivity-LoRA consistently outperforms LoRA across different rank budget scenarios on both datasets. On the SST-2 dataset, as the rank budget increases, the model performance exhibits an overall upward trend, with a notable improvement at $r = 5$ and a peak at $r = 8$. On the CoLA dataset, however, as the rank budget increases from 4, the model performance for the baseline method LoRA decreases after fine-tuning, while Sensitivity-LoRA initially declines slightly before increasing again. Despite some fluctuations, Sensitivity-LoRA consistently outperforms LoRA.

4.6 Case study

Figure 5 illustrates the performance of the GPT-2 Large and RoBERTa-base models, fine-tuned using



(a) SST-2 (b) CoLA

Figure 4: The evaluation metric changes for RoBERTa-base fine-tuned on the SST-2 and CoLA datasets under different rank budget scenarios for Sensitivity-LoRA and LoRA.

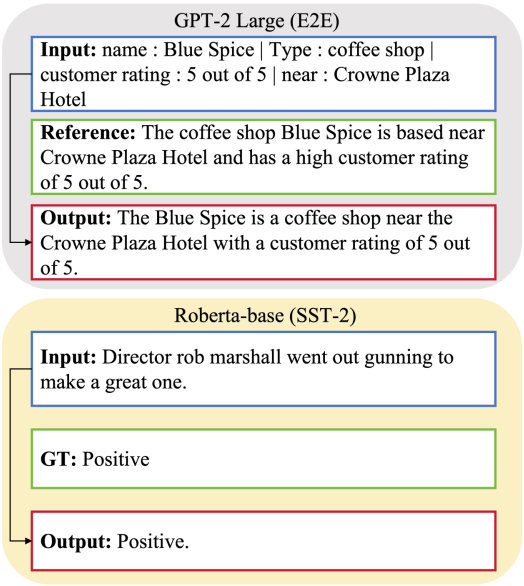


Figure 5: The Case Study of the GPT-2 Large and RoBERTa-base models. The blue boxes represent the input test data, the green boxes indicate the reference text or ground truth output, and the red boxes highlight the model's actual output.

the dynamic rank allocation method (Sensitivity-LoRA) on the E2E and SST-2 datasets. For the E2E dataset, the output of the GPT-2 Large model retains the input information while generating fluent and grammatically correct natural language text that closely aligns with the reference text. This suggests that the model effectively processes structured input data and excels in generating accurate and fluent natural language descriptions. For the SST-2 dataset, the RoBERTa-base model demonstrates strong performance in sentiment classification tasks, accurately classifying input text as "Positive". This outcome validates the efficacy of the Sensitivity-LoRA method in improving model performance on text classification tasks.

5 Conclusion

In this work, we introduce Sensitivity-LoRA, a method that efficiently allocates ranks to weight matrices based on their sensitivity, without a significant computational burden. Sensitivity-LoRA first performs sensitivity detection by analyzing both global and local sensitivities. It utilizes the second-order derivatives (Hessian matrix) of the loss function to accurately capture parameter sensitivity. Next, it optimizes rank allocation by aggregating global and local sensitivities, ensuring a comprehensive and fair evaluation metric. Extensive experiments consistently demonstrate the efficiency, effectiveness and stability of our method.

6 Limitations

In this study, we opt for predefined parameters $(\gamma_1, \gamma_2, etc.)$ when integrating global metrics and local metrics. While our approach effectively guides the rank allocation process, it does not guarantee an optimal solution for fine-tuning. Although dynamically adjusting parameters through heuristic learning could be a potential solution, it may significantly extend training time and increase resource consumption. Given that our core design principle is to maintain a low computational load, we ultimately decide to use predefined parameters to ensure the efficiency and simplicity of the system. In the future, we aim to explore an effective

solution for dynamically modifying predefined parameters.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

G. Ding et al. 2022. Efficient fine-tuning for resource-constrained systems. *Proceedings of the Machine Learning Conference*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.

N. Houlsby, A. Giurgiu, S. Jastrzebski, et al. 2019. Parameter-efficient transfer learning for nlp. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019:279–285.

E. J. Hu, Y. Shen, P. Wallis, et al. 2022a. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations (ICLR)*.

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

Yahao Hu, Yifei Xie, Tianfeng Wang, Man Chen, and Zhisong Pan. 2023. Structure-aware low-rank adaptation for parameter-efficient fine-tuning. *Mathematics*, 11(20):4317.

E. Huang et al. 2023. Evaluating large language models in complex scenarios. *Journal of Computational Linguistics*.

B. Lester, R. Al-Rfou, and N. Constant. 2021. The power of scale for parameter-efficient prompt tuning. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021:3045–3061.

C. Li et al. 2023. Fine-tuning techniques for efficient model adaptation. *AI Research Journal*.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*.

X. Li and P. Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation tasks. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021:4582–4597.

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

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd annual meeting of the association for computational linguistics (ACL-04)*, pages 605–612.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364.

Zequan Liu, Jiawen Lyn, Wei Zhu, Xing Tian, and Yvette Graham. 2024. Alora: Allocating low-rank adaptation for fine-tuning large language models. *arXiv preprint arXiv:2403.16187*.

Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. 2024. Dora: Enhancing parameter-efficient fine-tuning with dynamic rank distribution. *arXiv preprint arXiv:2405.17357*.

Jekaterina Novikova, Ondrej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Saarbrücken, Germany. ArXiv:1706.09254.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

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. *Advances in neural information processing systems*, 32.

A. Qin et al. 2023. Advances in state-of-the-art natural language processing. *Journal of NLP Research*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2022. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Alex Wang. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

F. Wang et al. 2023. Practical applications of llms in specialized domains. *Specialized AI Applications*.

A Warstadt. 2019. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Thomas Wolf. 2020. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

E. Zaken, Y. Goldberg, and S. Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformers. *Transactions of the Association for Computational Linguistics (TACL)*, 10:1–16.

D. Zhang et al. 2023a. Parameter-efficient fine-tuning methods for llms. *Journal of Machine Learning Research*.

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

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

Ruiyi Zhang, Rushi Qiang, Sai Ashish Somayajula, and Pengtao Xie. 2024. Autolora: Automatically tuning matrix ranks in low-rank adaptation based on meta learning. *arXiv preprint arXiv:2403.09113*.

B. Zhu et al. 2023a. Expanding frontiers in large language models. *AI Frontier Research*.

B. Zhu et al. 2023b. Large language models: Progress and applications. *Advances in NLP*.