Neural Networks 194 (2026) 108095

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Full Length Article

La-LoRA: Parameter-efficient fine-tuning with layer-wise adaptive low-rank adaptation

Jiancheng Gu [a,*], Jiabin Yuan [a,*], Jiyuan Cai [b], Xianfa Zhou [a], Lili Fan [c]

[a] School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China
[b] AI Core Dev., Lenovo IDG MBG, Nanjing, 100085, China
[c] School of Computer Engineering, Jinling Institute of Technology, Nanjing, 211169, China

ARTICLE INFO

Keywords:
Large language models
Parameter-efficient fine-tuning
Computational efficiency
Low-rank adaptation

ABSTRACT

Parameter-efficient fine-tuning (PEFT) has emerged as a critical paradigm for adapting large pre-trained models to downstream tasks, offering a balance between computational efficiency and model performance. Among these methods, Low-Rank Adaptation (LoRA) has gained significant popularity due to its efficiency; it freezes the pre-trained weights and decomposes the incremental matrices into two trainable low-rank matrices. However, a critical limitation of LoRA lies in its uniform rank assignment across all layers, which fails to account for the heterogeneous importance of different layers in contributing to task performance, potentially resulting in suboptimal adaptation. To address this limitation, we propose Layer-wise Adaptive Low-Rank Adaptation (La-LoRA), a novel approach that dynamically allocates rank to each layer based on Dynamic Contribution-Driven Parameter Budget (DCDPB) and Truncated Norm Weighted Dynamic Rank Allocation (TNW-DRA) during training. By treating each layer as an independent unit and progressively adjusting its rank allocation, La-LoRA ensures optimal model performance while maintaining computational efficiency and adapting to the complexity of diverse tasks. We conducted extensive experiments across multiple tasks and models to evaluate the effectiveness of La-LoRA. The results demonstrate that La-LoRA consistently outperforms existing benchmarks, validating its effectiveness in diverse scenarios.

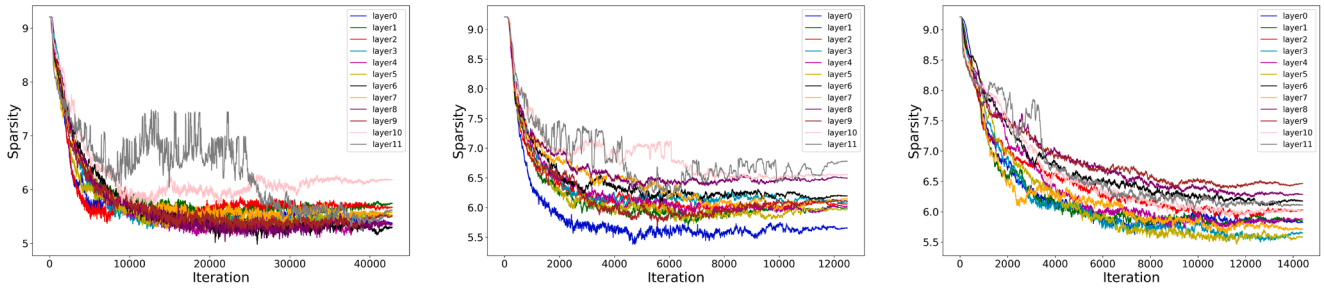
1. Introduction

Large language models (LLMs), with their extensive parameters and intricate architectures, exhibit remarkable task generalization capabilities after being trained on vast datasets. They have demonstrated exceptional performance across a wide range of natural language processing (NLP) tasks (Achiam et al., 2023; Bi et al., 2024; Brown et al., 2020; Mars, 2022; Team et al., 2024; Touvron et al., 2023). However, despite the remarkable performance of LLMs on general tasks, their effectiveness remains limited when applied to domain-specific tasks. The fine-tuning of LLMs for downstream tasks has emerged as a prominent and widely adopted paradigm, drawing significant attention from the research community. However, traditional full-parameter fine-tuning methods require the adjustment of all model parameters, leading to a sharp increase in computational costs and storage demands. Coupled with the growing trend towards increasingly larger models, this escalating demand for computational resources has raised significant concerns regarding scalability and accessibility, thereby limiting their applicability in resource-constrained environments. Similar challenges in balancing model performance and resource efficiency have been observed in different application areas; for example, Vaigandla and Siddoju (2025) presents demonstrates the application of neural networks in orthogonal frequency-division multiplexing (OFDM) / fifth-generation (5G) systems to reduce network costs and system complexity. Lateef (2024) reviews machine and deep learning approaches for enhancing cancer prediction and risk assessment. In bioinformatics with limited computational resources, making efficient use of available resources is important.

In response to this challenge, several parameter-efficient fine-tuning (PEFT) methods have emerged, aiming to optimize LLMs for downstream tasks while reducing resource consumption. These methods achieve, or even surpass, the performance of full parameter fine-tuning by training only a small subset of the parameters. PEFT methods can be broadly categorized into three types: Addition-based, Selection-based, and Reparameterization-based methods (Lialin et al., 2023). Addition-based methods involve introducing additional parameters or layers and training only the newly introduced components. For example, Prefix tuning (Li & Liang, 2021) and Prompt tuning (Lester et al., 2021) introduce supplementary trainable prefix tokens, which are attached either to

* Corresponding authors.
E-mail addresses: jianchenggu@nuaa.edu.cn (J. Gu), jbyuan@nuaa.edu.cn (J. Yuan), caijy6@lenovo.com (J. Cai), xfzhou@nuaa.edu.cn (X. Zhou), lilyfan@jit.edu.cn (L. Fan).

https://doi.org/10.1016/j.neunet.2025.108095
Received 27 March 2025; Received in revised form 12 August 2025; Accepted 4 September 2025
Available online 10 September 2025



(a) CoLA (b) RTE (c) STS-B

Fig. 1. The sparsity of the LoRA increment matrix in each layer based on the RoBERTa base model on the CoLA, RTE and STS-B datasets.

the input, or the hidden layers of the base model. However, these methods alter the original model structure and introduce additional cost during the inference phase. Selection-based methods achieve efficient fine-tuning by selectively choosing specific layers, parameters, or structures in the network. For instance, BitFit (Zaken et al., 2022) trains only the bias terms in the network, while Diff Pruning (Guo et al., 2021) extends the original pretrained parameters by learning a task-specific "difference" vector. Both methods significantly reduce the number of trainable parameters while maintaining performance. Reparametrization-based methods leverage low-rank representations to minimize the number of trainable parameters. For example, LoRA (Hu et al., 2021), as one of the currently most popular methods for fine-tuning, alleviates the drawbacks of the aforementioned approaches. It achieves this by freezing the pre-trained weights and introducing the product of low-rank matrices (i.e., W_{dw}, W_{up}) alongside the original weight matrix (i.e., W_0) as the only trainable parameters, approximating the gradient updates (i.e., ΔW) in the original matrix. Building upon the success of LoRA, several works have sought to further improve its efficiency and performance. Notably, AdaLoRA (Zhang et al., 2023b) represents the incremental matrix in the form of Singular Value Decomposition (SVD) and performs adaptive rank adjustment by pruning singular values based on their importance.

The Lottery Ticket Hypothesis (Frankle, 2018) suggests that within an over-parameterized neural network, there exists a smaller subnetwork that can be trained independently and performs comparably or even better than the full-scale model. During model training, not all layers contribute equally to the final performance. By assigning lower ranks to the layers that have less impact on overall performance, it is possible to reduce computational costs while maintaining model accuracy, thereby enhancing efficiency. However, assigning uniform ranks to these less-contributing layers may lead to overfitting, potentially undermining the model's capacity and degrading its generalization ability. Building upon this observation, it becomes evident that a static rank allocation, as employed in LoRA, may not be optimal for all layers. Some layers may warrant higher ranks due to their substantial impact on performance, while others may benefit from lower ranks to avoid unnecessary computational overhead.

Inspired by the aforementioned pioneering work, we embarked on a preliminary exploration with LoRA to dissect the intricacies of network training through the lens of sparsity. Our study is anchored in the analysis of three prominent natural language processing (NLP) datasets: CoLA, RTE, and STS-B, with RoBERTa (Liu, 2019) as the backbone model. The journey began with a pilot analysis, where we sought to quantify the sparsity within the layers of a neural network during the training process. To this end, we defined a sparsity metric to quantify the proportion of near-zero elements within the layers of the network. Specifically, for a layer with a weight matrix P of dimensions m × n, our sparsity metric is calculated as:

$$spa = \log \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{1}\left[P_{\mathrm{m,n}} < \epsilon\right]}{\epsilon \times m \times n} \tag{1}$$

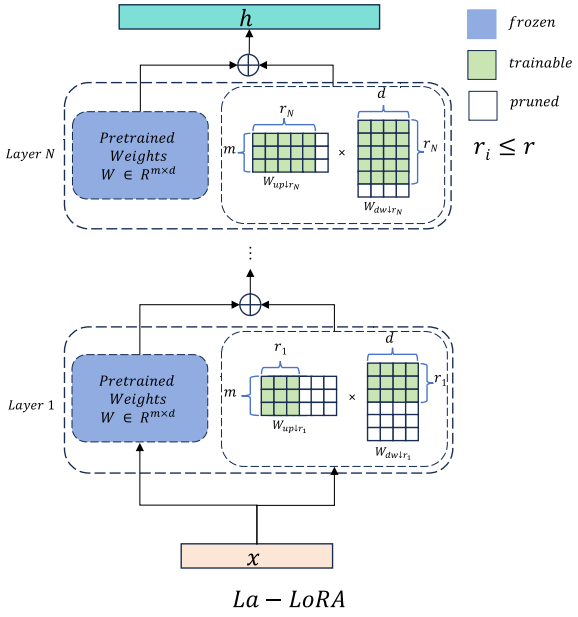


Fig. 2. La-LoRA adaptively allocates ranks ($r_i \leq r$) across layers, with some components pruned to optimize parameter efficiency and performance.

where ϵ refers to a tiny value.[1] As illustrated in Fig. 1, each layer of the network follows a unique trajectory as its sparsity evolves, culminating in a distinct convergence point. This divergence in sparsity profiles suggests that the layers are not monolithic entities, but have different roles and capacities for information integration during training. Intuitively, a more nuanced, layer-specific strategy may be necessary to fully harness the potential of LoRA and similar techniques.

Motivated by the above, this paper take the magnitude of layer-specific rank into consideration, and proposes a more tailored adaptation of rank with respect to LoRA, namely Layer-wise Adaptive Low-Rank Adaptation (La-LoRA). Instead of assigning a fixed rank to layers, La-LoRA treats each layer as the minimum unit, evaluating its contribution to the overall performance. Taking the current available rank into account, it adaptively allocates a more desirable rank to each layer according to its significance. During training, La-LoRA controls model complexity, allowing the model to self-adapt to different tasks, as shown in Fig. 2. The main contributions can be summarized as:

- We introduce LA-LoRA, an intuitive and effective PEFT method that allocates ranks to layers based on Dynamic Contribution-Driven Parameter Budget (DCDPB) and Truncated Norm Weighted Dynamic

[1] $1e-4$ is used in this paper.

Rank Allocation (TNW-DRA) during training and do not introduce additional cost during the inference phase.
- We propose the Dynamic Contribution-Driven Parameter Budget (DCDPB), a method that dynamically allocates parameter budgets based on each layer's marginal contribution potential during training. In the early stages, fewer budgets are allocated to learn foundational features, while in later stages, budgets are increased to capture more complex features. This step-by-step allocation strategy enables the model to progressively learn from simple to complex patterns, optimizing resource utilization.
- We propose the Truncated Norm Weighted Dynamic Rank Allocation (TNW-DRA). This method involves truncating the matrices of layers with lower contributions, thereby avoiding overfitting while optimizing computational efficiency. To more accurately assess each layer's actual contribution, we introduce a correction factor to adjust the contribution values of truncated matrices.
- We evaluate La-LoRA across various tasks, showing improved performance over existing benchmarks in our experiments.

2. Related work

2.1. Lottery ticket hypothesis

The Lottery Ticket Hypothesis has garnered considerable attention in the deep learning community for its implications on network training efficiency and pruning strategies. It proposes the intriguing notion that a sparse, high-performing sub-network, or "winning ticket", is embedded within a larger, randomly initialized network, and can be efficiently trained with minimal loss in performance. Gohil et al. (2019) extends the hypothesis by examining the transferability of winning tickets across various datasets and optimizers. This work unveils the potential for pre-generating a limited set of tickets for reuse in multiple tasks, streamlining the training process. Building upon this, Yu et al. (2020) delves into the presence of winning tickets in domains ranging from supervised learning to both reinforcement learning and natural language processing, signifying the hypothesis as a pervasive characteristic of deep neural networks. It points toward the possibility of simplifying the architecture and training of complex models, such as attention-based language processors, for more practical deployment. Subsequent research by Tian (2019) dispells several mathematical assumptions, especially noting that weights with minor initialization advantages are more likely to constitute winning tickets after training.

2.2. Parameter-efficient fine-tuning (PEFT)

Full fine-tuning(FT), which involves the comprehensive update of all parameters in pre-trained language models (Devlin, 2018; Radford et al., 2019), is crucial for tailoring models to specific downstream tasks in NLP, such as text classification and question answering. However, this approach presents challenges for large-scale models (Brown et al., 2020; Chung et al., 2024), due to the substantial computational and storage demands and the risk of catastrophic forgetting (Luo et al., 2023), particularly when there is a significant divergence between pre-training and target task objectives. PEFT addresses these challenges by targeting only a subset of model parameters for optimization. This selective parameter update strategy reduces the computational burden and storage requirements, while also preserving the pre-trained knowledge to a greater extent, thus offering a more efficient and scalable fine-tuning solution for LLMs (Li & Liang, 2021; Rücklé et al., 2021a). Adapter Modules (Houlsby et al., 2019) added a small network structure named Adapter between the layers of the pre-trained model. During training, only the Adapter parameters are adjusted, leaving the original model parameters unchanged. Prefix-Tuning (Li & Liang, 2021) adjusts the input by optimizing a sequence of continuous prompt vectors. This approach minimizes parameter updates and performs remarkably well in generation tasks. Prompt-Tuning (Lester et al., 2021) further refined this idea by fine tuning lightweight prompt embeddings, it enhanced the adaptability of LLMs. AdapterDrop (Rücklé et al., 2021b) improved efficiency by dynamically discarding some Adapter modules.

2.3. LoRA and its variants

Building on previous research, LoRA (Hu et al., 2021) introduces a new approach that updates weight increments using low-rank matrix decomposition. This method significantly reduces the number of parameters, computational complexity, and hardware requirements, while maintaining the same inference time. It marks an important step forward in parameter-efficient fine-tuning (PEFT). AdaLoRA (Zhang et al., 2023b) adapts ranks based on singular values but tends to allocate higher-than-necessary ranks in the initial stages, resulting in wasted computational resources. LoRA-FA (Zhang et al., 2023a) freezes the lower projection matrix A and only updates the upper projection matrix B, cutting down on memory use, which works well for low-resource setups. DyLoRA (Valipour et al., 2023) randomly assigns ranks to each layer during training and trains with multiple rank options at once. VeRA (Kopiczko et al., 2023) uses shared random matrices A and B, plus small scaling vectors for each layer, reducing the parameter count a lot compared to LoRA while still holding decent performance. DoRA (Liu et al., 2025) splits weights into magnitude and direction parts for tuning, making model optimization more precise. PiSSA (Meng et al., 2024) applies SVD to initialize the trainable matrices A and B with the principal singular values of W, while the residual matrix W_{res} is initialized with the residual singular values and kept frozen during fine-tuning. MiLoRA (Wang et al., 2025) fine-tunes only the minor singular components of the weight matrix while freezing the major singular components. RandLoRA (Albert et al., 2025) performs full-rank updates by learning linear combinations of fixed random matrices, optimizing only diagonal scaling matrices to minimize trainable parameters.

3. Method

In this paper, we propose Layer-wise Adaptive Low-Rank Adaptation (LA-LoRA), a method that dynamically allocates intrinsic ranks to layers during fine-tuning, considering factors like matrix truncation, relative fidelity rate of the increment matrix f, Dynamic Contribution-Driven Parameter Budget (DCDPB), and each layer's contribution. The overall algorithm flow is shown in Algorithm 1.

Algorithm 1 La-LoRA.

Require: maximum intrinsic rank r, total iterations T, the rank of the i-th layer r_i (initially, $r_i = 1$), hyperparameter γ

for $t = 1$ to T **do**
  $W_{dw\downarrow r_i} \Leftarrow W_{dw}[1 : r_i, :]$
  $W_{up\downarrow r_i} \Leftarrow W_{up}[:, 1 : r_i]$
  Calculate the value of each layer in the current step by Eq. 7.
  Calculate the growth rate of each layer relative to previous step by Eq. 6.
  Calculate the current total allocatable rank based on the overall growth rate by Eq. 5.
  Reasonably allocate the appropriate rank to each layer according to each layer's contribution to the overall contribution and the total distributable rank by Eqs. 8 and 9 .
  Update the rank of each layer.
end for

3.1. Parameterization

Before delving into our proposed method, let's briefly review the conventional approach of LoRA and its underlying principles. A standard

LoRA layer can be represented by the following equation:

$$\begin{aligned} h^{t+1} &= W_0 x + \Delta W^t x \\ &= W_0 x + \frac{\alpha}{r} W_{up}^t W_{dw}^t x \end{aligned} \tag{2}$$

where $W_0 \in R^{m\times d}$ represents the weights of the pre-trained model, which remain frozen throughout the fine-tuning process at every step. α is a constant scale hyper-parameter. Only the LoRA modules(i.e., $W_{dw}^t \in R^{r\times d}$, $W_{up}^t \in R^{m\times r}$) are trainable. t represents the fine-tuning steps. We dynamically allocate an intrinsic rank $r_i' \in [1, r]$, $i \in [1, L]$ for the i-th layer at each fine-tuning step. The first r' rows (or columns) from the original matrix are retained, while the remaining parts are temporarily pruned. The truncated matrices are denoted as $W_{dw\downarrow r'}$ and $W_{up\downarrow r'}$.

$$\begin{aligned} W_{dw\downarrow r'} &= W_{dw}[1 : r', :] \\ W_{up\downarrow r'} &= W_{up}[:, 1 : r'] \end{aligned} \tag{3}$$

Then, the output of each layer is at each fine-tuning step as follows:

$$\begin{aligned} h^{t+1} &= W_0 x + \Delta W^t x \\ &= W_0 x + \frac{\alpha}{r} W_{up\downarrow r'}^t W_{dw\downarrow r'}^t x \end{aligned} \tag{4}$$

3.2. Dynamic contribution-driven parameter budget

The complexity requirements of a model vary not only across different tasks and datasets but also evolve over the course of training. Specifically, in the early stages of training, fewer parameters are needed as the model begins to learn basic patterns; however, as training progresses, more parameters become necessary to capture increasingly complex features. To address this dynamic variation and optimize the use of limited parameter resources, we propose Dynamic Contribution-Driven Parameter Budget (DCDPB). This method adaptively adjusts rank allocation during training, enabling the model to dynamically modulate its complexity based on both the evolving demands of the training process and the specific characteristics of the dataset. In this way, DCDPB ensures efficient resource utilization while maintaining or even enhancing model performance.

Rather than directly setting the DCDPB to its maximum value, it is progressively adjusted towards this maximum through an iterative process. This adjustment is guided by the growth rate of each layer's value, s_i, ensuring that resources are allocated based on their evolving importance during training. The growth rate of each layer's value, s_i, is calculated relative to its previous state, and the average of these growth rates across all layers is then determined to obtain the cumulative growth rate of the DCDPB. This adaptive adjustment mechanism optimizes the utilization of the parameter budget. The calculation for the current allocatable rank (DCDPB, with an initial value of $1L$) at the fine-tuning step is as follows:

$$\begin{aligned} DCDPB^t &= DCDPB^{t-1}(1 + \overline{IR^t}) \\ \overline{IR^t} &= \gamma \frac{1}{L} \sum_{i=1}^{L} IR_i^t \\ DCDPB^t &\in [1L, rL] \end{aligned} \tag{5}$$

where L represents the total number of layers in the model, IR_i^t denotes the rate of increase of the i-th layer's value, s_i, during the t-th step of the fine-tuning process. γ represents the amplification factor, which is used to control the growth rate of the DCDPB.

To compute the growth rate of each layer's value, s_i, we choose the Frobenius norm as the measure of its contribution and quantify the change in this norm before and after the increment. The growth rate captures the marginal potential of each layer to enhance overall model performance during training, reflecting its capacity for greater performance gains at the current stage. By calculating the overall growth rate, we can obtain a global perspective that reflects the changing trends in the model's overall complexity, thereby guiding the allocation of resources at a global scale. The specific calculation formula is as follows:

$$IR_i^t = \frac{s_i^t - s_i^{t-1}}{s_i^{t-1}} \tag{6}$$

where

$$\begin{aligned} s_i^t &= \frac{1}{r_i^t} \sum (\|W_{up\downarrow r_i^t}^t\|_F + \|W_{dw\downarrow r_i^t}^t\|_F) \\ &\text{s.t.}, W_{up\downarrow r_i^t}^t \in W^{ft}, W_{dw\downarrow r_i^t}^t \in W^{ft} \end{aligned} \tag{7}$$

The sum of the Frobenius norms of the matrix modules that require fine-tuning in each layer is used as the value of the current layer. W^{ft} represents the matrix module that requires fine-tuning, including the query, key, and value matrices within the transformer architecture. In the experiments, the method was applied exclusively to the query and value projection matrices in each self-attention module.

3.3. Truncated norm weighted dynamic rank allocation

Upon determining the DCDPB during training, an allocation strategy is required to accomplish the assignment of ranks. To this end, we propose the Truncated Norm Weighted Dynamic Rank Allocation (TNWDRA) to assign higher intrinsic ranks to layers that are critical for improving overall performance, while minimizing the rank of those with less significant contributions. This approach achieves a strategic balance between resource allocation and model efficiency. The specific calculation formula for the overall allocation strategy is as follows:

$$\begin{aligned} &[r_1^{t+1}, r_2^{t+1}, \ldots, r_L^{t+1}] \\ &= Softmax([c_1^t, c_2^t, \ldots, c_L^t]) DCDPB^t \end{aligned} \tag{8}$$

where c_i^t represents the contribution of the i-th layer at t-th fine-tuning step. We calculate the Frobenius norm of the truncated matrix to measure its contribution, as it provides a more comprehensive reflection of the matrix's overall "energy." By accounting for the sum of squares of all retained singular values, the Frobenius norm not only enables a more effective assessment of the truncated matrix's overall contribution but also allows us to better integrate the effective information from W_{up} and W_{dw}. In the process of computation, we perform calculations on the truncated matrix. Directly using the Frobenius norm of the truncated matrix as the contribution value may lead to underestimation or overestimation of the importance of certain layers, as the truncated matrix, while retaining primary features, loses some detailed information. To mitigate this bias, we calculate the ratio of the Frobenius norm of the truncated matrix to that of the original matrix and multiply it back to the Frobenius norm of the truncated matrix. This approach provides a more accurate representation of the true contribution of each layer. The contribution of each layer and the fidelity of the matrix f are calculated as follows:

$$\begin{aligned} c_i^t &= \sum (f_{dw}^t \|W_{up\downarrow r_i^t}^t\|_F + f_{up}^t \|W_{dw\downarrow r_i^t}^t\|_F) \\ &\text{s.t.}, W_{up\downarrow r_i^t}^t \in W^{ft}, W_{dw\downarrow r_i^t}^t \in W^{ft} \end{aligned} \tag{9}$$

where

$$\begin{aligned} f_{dw}^t &= \frac{\|W_{dw\downarrow r_i^{t-1}}^t\|_F}{\|W_{dw\downarrow r}^t\|_F} \\ f_{up}^t &= \frac{\|W_{up\downarrow r_i^{t-1}}^t\|_F}{\|W_{up\downarrow r}^t\|_F} \end{aligned} \tag{10}$$

4. Experiments

We describe the experiments conducted to validate our proposed method across a diverse set of tasks, including natural language understanding (NLU), question answering (QA), summarization, image

Table 1
Summary of the GLUE benchmark.

Task Name	Task Type	Data Size (Train/Dev/Test)	Metric	Task Description
CoLA	Single-sentence	8.55k/1.04k/1.06k	Matthews Corr	Judge if a sentence is grammatically correct
SST-2	Single-sentence	67.3k/872/1.82k	Accuracy	Classify sentiment (positive/negative)
MRPC	Sentence-pair	3.67k/408/1.73k	Accuracy/F1	Determine if two sentences are paraphrases
STS-B	Sentence-pair	5.75k/1.5k/1.38k	Pearson/Spearman Corr	Assess semantic similarity (0–5 scale)
QNLI	Sentence-pair	105k/5.46k/5.46k	Accuracy	Determine if a sentence answers a question
RTE	Sentence-pair	2.49k/277/3k	Accuracy	Classify entailment relation

Table 2
Summary of the SQuAd dataset.

	Data Size (Train/Dev)	Metric	Task Description
SQuAD v1.1	87.6k/10.6k	EM/F1	Extract answer spans from Wikipedia articles
SQuAD v2.0	130k/11.9k	EM/F1	Extract answer spans or identify unanswerable questions

Table 3
Summary of the CNN/DailyMail dataset and image datasets.

	Data Size (Train/Dev/Test)	Metric	Task Description
CNN/DailyMail	287k/13.4k/11.5k	ROUGE-1/2/L	Generate summaries from news articles
CIFAR-100	50k/-/10k	Accuracy	Classify 100 categories of small images
Food-101	75.8k/25.3k/-	Accuracy	Classify 101 categories of food images
Flowers-102	1020/1020/6149	Accuracy	Classify 102 categories of flower images
RESISC45	18.9k/6.3k/6.3k	Accuracy	Classify 45 categories of remote sensing scenes

Table 4
Summary of the commonsense reasoning.

	Data Size (Train/Dev/Test)	Metric	Task Description
BoolQ	9.43k/3.27k/-	Accuracy	Answer true/false questions
PIQA	1.6k/2k/3k	Accuracy	Physical commonsense reasoning
SIQA	33.4k/1.95k/-	Accuracy	Social commonsense reasoning
HellaSwag	39.9k/10k/10k	Accuracy	Sentence completion reasoning
WinoGrande	40.4k/1.3k/1.8k	Accuracy	Pronoun resolution
ARC-e	2.25k/570/2.38k	Accuracy	Commonsense reasoning (easy)
ARC-c	1.12k/299/1.17k	Accuracy	Commonsense reasoning (challenging)
OBQA	4.96k/500/500	Accuracy	Open-book commonsense QA

classification, and commonsense reasoning. Our experiments leverage the *Huggingface Transformers* code-base. Our experiments were conducted on a machine equipped with an NVIDIA GeForce RTX 4090 GPU.

4.1. Datasets

To validate the effectiveness of the La-LoRA method on natural language understanding tasks, we chose the GLUE (Wang et al., 2018) dataset and the RoBERTa model for our experiments. The GLUE dataset includes nine tasks covering various types such as single-sentence classification, sentence-pair classification, and question answering. We selected the six most popular tasks from this dataset for our experiments. The tasks and data sizes for each dataset are shown in Table 1.

To evaluate La-LoRA's performance on Question Answering tasks, we conducted experiments using the SQuAD v1.1 (Rajpurkar et al., 2016) and v2.0 (Rajpurkar et al., 2018) benchmarks with the DeBERTa model. The SQuAD v1.1 dataset contains 536 Wikipedia articles, where crowdworkers annotated question-answer pairs. In this task, given a passage and a question, the model simply extracts the relevant text span as the answer. SQuAD v2.0 builds upon v1.1 by introducing unanswerable questions, forcing the model to first determine answerability before either extracting answers or responding "unanswerable", which significantly increases the challenge. Detailed dataset statistics are provided in Table 2.

To validate La-LoRA's effectiveness on text summarization tasks, we used the CNN/DailyMail (Hermann et al., 2015) dataset and the T5 model. CNN/DailyMail is a widely-used benchmark dataset for text summarization, based on online news articles from CNN and DailyMail. It includes about 310,000 documents, each paired with a short summary written by editors. This dataset is commonly used to train and evaluate models' summarization capabilities, supporting both extractive and abstractive methods, with performance typically measured using ROUGE metrics. For evaluating La-LoRA on image classification tasks, we selected the VIT model and four popular datasets: CIFAR-100 (Krizhevsky et al., 2009), Food-101 (Bossard et al., 2014), Flowers-102 (Nilsback & Zisserman, 2008), and RESISC45 (Cheng et al., 2017). CIFAR-100 contains 100 classes of small images (32x32 pixels), totaling 60,000 images, ideal for quick classification performance testing. Food-101 covers 101 food categories with 101,000 images in total, suitable for food recognition and noise data handling. Flowers-102 includes 102 types of flowers, with over 8000 images, challenging fine-grained classification abilities. RESISC45 provides 45 classes of remote sensing scenes, with 31,500 images at 256x256 pixels, useful for assessing remote sensing image classification. These datasets are widely used to test model classification accuracy and generalization capabilities. Dataset details can be found in Table 3.

We also tested the model on 8 commonsense reasoning tasks using LLaMA-1 and LLaMA-2: BooIQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-e/ARC-c (Clark et al., 2018), and

Table 5
Result with RoBERTa base on GLUE. Optimal performance results for each dataset are indicated in **bold** font. We report results on development sets, Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy metrics for the remaining tasks.

Method	# Trainable Parameters	Accuracy SST-2	Accuracy MRPC	Mathews CoLA	Accuracy QNLI	Accuracy RTE	Person STS-B	Avg.
Full FT	125M	94.8	**90.2**	63.6	92.8	78.7	**91.2**	85.2
BitFit	0.1M	94.4	89.1	61.2	91.0	79.6	90.6	84.3
$Adpt^D$	0.3M	94.2	88.5	60.8	**93.1**	71.5	89.7	83.0
$Adpt^D$	0.9M	94.7	88.4	62.6	93.0	75.9	90.3	84.2
$Adpt^H$	0.3M	93.5	88.6	61.8	92.5	78.6	90.9	84.3
$Adpt^P$	0.3M	93.2	88.7	62.9	92.6	79.1	90.4	84.5
LoRA	0.3M	94.0	88.7	63.4	92.4	78.6	90.5	84.6
DyLoRA	0.3M	94.4	87.3	59.5	93.0	77.6	91.1	83.8
LoRA-FA	1.8M	94.8	90.0	63.6	92.5	67.9	89.6	83.1
AdaLoRA	0.3M	94.8	88.8	63.2	92.9	79.0	91.1	85.0
DoRA	0.3M	93.5	87.6	64.2	91.0	79.7	91.0	84.5
PISSA	0.3M	94.6	87.6	63.7	92.4	79.6	90.6	84.8
MiLoRA	0.1M	94.3	87.7	64.2	92.7	80.2	90.1	84.9
RandLoRA	0.8M	93.8	87.6	64.7	92.0	79.0	90.9	84.7
La-LoRA(ours)	0.3M	**94.8**	88.0	**65.0**	92.9	**80.4**	91.0	**85.4**

OBQA (Mihaylov et al., 2018). BooIQ checks true-or-false answers, PIQA and SIQA focus on physical and social commonsense, HellaSwag and WinoGrande test sentence completion and pronoun resolution, ARC-e and ARC-c are multiple-choice reasoning tasks with easy and challenging levels, and OBQA involves open-book question answering. These tasks cover social, physical, and logical commonsense reasoning, providing a solid overall view of the model's performance. Detailed results are shown in Table 4.

4.2. Results

4.2.1. Natural language understanding

Across the GLUE benchmark, we compare our method with baselines like LoRA, DyLoRA, LoRA-FA, and AdaLoRA. We did not employ the MLNI technique, which involves using the MLNI checkpoint in place of the pretrained weights, to improve the model's performance. For the experimental setup, we followed the configuration of LoRA (Hu et al., 2021). We selected five random seeds and reported the average results across these seeds. As shown in Table 5, our method shows improved performance over others, achieving 65.0 % on CoLA (2.2 % higher than the best baseline) and 80.4 % on RTE (7.9 % higher than LoRA). Specifically, La-LoRA outperforms full fine-tuning on the GLUE dataset while using only 0.24 % of the parameters, emphasizing La-LoRA's efficiency in maximizing performance with a minimal parameter budget. Additionally, with the same parameter budget, La-LoRA achieves a 1.8 % performance improvement over LoRA. La-LoRA extends LoRA without increasing the parameter budget.

4.2.2. Question answering

Table 6 illustrates that La-LoRA consistently outperforms other methods across both the SQuADv1.1 and SQuADv2.0 datasets, especially when the parameter budget is constrained. At 0.08 % of trainable parameters, La-LoRA achieves an EM/F1 score of 87.2/93.3 on SQuADv1.1 and 83.5/86.7 on SQuADv2.0, surpassing both LoRA and DyLoRA. This trend continues at 0.16 % and 0.32 % parameter budgets, where La-LoRA maintains its advantage, demonstrating that its adaptive rank allocation leads to superior fine-tuning performance. These results confirm that La-LoRA effectively balances model complexity and parameter efficiency.

4.2.3. Summarization and image classification

From Tables 7 and 8, it can be observed that La-LoRA shows improved performance over LoRA and even surpasses the performance of full fine-tuning on most datasets. For instance, FT achieves the best performance on the R-2 metric of the CNN/DailyMail dataset, reaching 19.56, but La-LoRA outperforms in all other metrics. Compared to LoRA, La-LoRA consistently surpasses it across different ranks. This indicates that La-LoRA can better utilize parameters and control model complexity based on the dataset during training to improve overall performance. For a detailed analysis, see Section 5.5.

Table 6
Result with DeBERTaV3-base (He et al., 2020) on SQuADv1.1 and SQuADv2.0.We report Exact Math(EM)/F1 and '# Params' denotes the quantity of trainable parameters relative to full-parameter fine-tuning.The best results are in **bold**.

# Params	Method	SQuADv1.1	SQuADv2.0
100 %	FT	86.0/92.7	**85.4/88.4**
0.08 %	DyLoRA	86.4/92.8	82.5/85.8
	LoRA	86.6/93.0	83.2/86.4
	La-LoRA	**87.2/93.3**	**83.5/86.7**
0.16 %	DyLoRA	86.8/93.0	83.3/86.5
	LoRA	86.6/92.8	83.0/86.2
	La-LoRA	**87.4/93.4**	**83.7/86.8**
0.32 %	DyLoRA	85.8/92.6	82.8/86.3
	LoRA	86.2/92.8	83.6/86.8
	La-LoRA	**86.5/93.0**	**84.1/87.2**

Table 7
Result with T5-small on CNN/DailyMail. We report R-1/2/L. The best results are in **bold**.

Rank	Method	CNN/DailyMail
-	FT	41.12/**19.56**/38.35
4	LoRA	41.03/18.76/38.26
	La-LoRA	**41.14/18.79/38.36**
8	LoRA	41.19/18.90/38.40
	La-LoRA	**41.36/18.97/38.59**
16	LoRA	41.15/18.84/38.36
	La-LoRA	**41.27/18.91/38.47**

Table 8
Result with VIT-B/L on different image classification datasets. The best results are in **bold**.

	Method	# Parmas	CIFAR100	Food101	Flowers102	RESISC45
VIT-B	FT	100 %	92.06	**91.12**	98.90	96.08
	LoRA	0.43 %	92.09	90.40	98.92	95.67
	La-LoRA	0.43 %	**92.65**	90.51	**99.41**	**96.72**
VIT-L	FT	100 %	**93.29**	87.13	98.90	95.7
	LoRA	0.29 %	92.75	87.10	98.92	94.52
	La-LoRA	0.29 %	92.97	**87.21**	**99.12**	**96.23**

4.2.4. Commonsense reasoning

The commonsense reasoning task comprises eight subtasks, each with its corresponding training and test sets. Following the experimental setup of DoRA (Liu et al., 2025), we only adjusted the batch size due to GPU memory constraints. We compared La-LoRA with several baseline methods, including Prompt Learning (Prefix) (Li & Liang, 2021), Series Adapter (Series) (Houlsby et al., 2019), and Parallel Adapter (Parallel) (He et al., 2021) on LLaMA. As shown in Table 9, La-LoRA is generally superior to the state-of-the-art method DoRA. From the perspective of parameter scale, GPT-3 and PaLM boast 175B and 540B parameters respectively, whereas LLaMA 7B and LLaMA2 7B contain a mere 7B parameters, indicating that the latter have a significant advantage in parameter efficiency. However, despite having far fewer parameters than GPT-3 and PaLM, LLaMA 7B and LLaMA2 7B, through the application of various PEFT techniques, are still able to achieve performance on multiple datasets that is close to or even surpasses that of the larger models. For the LLaMA 7B model, La-LoRA significantly outpaces other excellent methods. Compared to the state-of-the-art (SOTA) method DoRA, La-LoRA has improved accuracy by 1.6 %, nearly approaching the effectiveness of ChatGPT. Meanwhile, for the LLaMA2 7B model, La-LoRA has enhanced accuracy by 2.4 % over the SOTA method DoRA, even surpassing the accuracy level of ChatGPT. This result further demonstrates the effectiveness of the La-LoRA approach and highlights the substantial value of parameter-efficient fine-tuning (PEFT) in enhancing the performance of LLMs.

5. Analysis and discussion

5.1. Gradually increasing current allocable rank

To validate the effectiveness of gradually increasing the current allocatable rank (CAR), rather than directly setting it to the maximum value, rL, we fine-tuned the model on multiple datasets, with the results shown in Table 10.

Gradually increasing the CAR can effectively utilize parameters, ensuring that the model initially learns only the most important features to avoid overfitting. By controlling the expansion of capacity, this approach allows for an increase in rank with the progression of training to learn more complex patterns and prevent a decline in generalization performance due to excessively rapid capacity growth. This method also permits a step-by-step evaluation of the importance of each layer, assigning higher ranks based on actual contributions, thereby optimizing the use of ranks. The progressive increment strategy further helps save computational resources and memory during training, enabling more efficient resource allocation.

5.2. Matrix relative fidelity f

We use the Frobenius norm of the matrix to calculate each layer's contribution value, accounting for the expressive loss due to matrix truncation. We tested the effect of incorporating matrix fidelity, and the experimental results are shown in Table 11.

We use the Frobenius norm of a matrix to represent its contribution. Since recent studies (Lu et al., 2022; Peng et al., 2021; Schrimpf et al., 2021) have demonstrated the unexpected effectiveness of models that employ random weights and projections, directly using the Frobenius norm of the original initialized matrix may not accurately reflect its actual contribution. The truncated matrix, due to information loss, does not adequately represent its contribution. To address this issue, we multiply the Frobenius norm of the truncated matrix by its fidelity relative to the original initialization. This approach accounts for both the size of the truncated matrix and the retention of information, providing a more accurate assessment of its actual role in the model.

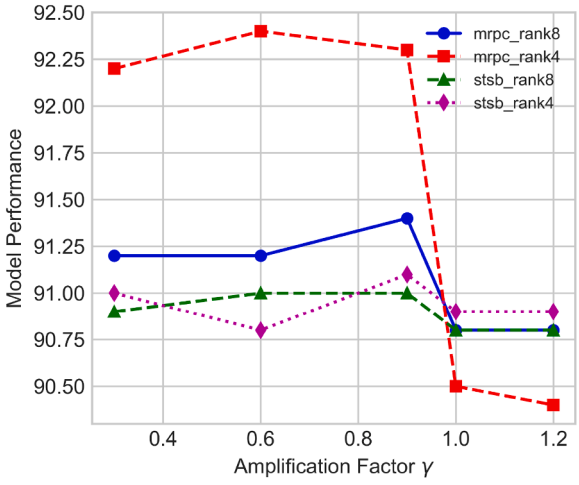


Fig. 3. The impact of different amplification factor.

5.3. Impact of amplification factor γ

We evaluated the impact of varying amplification factors γ on the model's performance using the MRPC and STS-B datasets. The results are presented in Fig. 3.

By introducing the amplification factors γ to control the rank growth rate, we can adjust the model's complexity, which is crucial for preventing overfitting and ensuring stable performance. This adjustment also optimizes resource utilization by aligning the model's capacity with task complexity, thereby reducing waste in simpler tasks while preserving expressiveness in more complex ones. Additionally, γ offers a flexible mechanism to adapt the model's complexity to varying task requirements, enhancing both its generalization ability and efficiency.

5.4. Correlation between sparsity and rank

We present an analysis based on pilot experiments conducted to assess the efficacy of our proposed method. The experiments were designed to train models on the CoLA, RTE, and STS-B datasets. Our analysis focuses on the final allocation ranks achieved by the models trained on these datasets and further examines the sparsity of the models' final weights, employing the computational approach outlined in the pilot study.

The results, as depicted in Fig. 4, demonstrate a consistent trend between the allocation ranks and the sparsity of the models' weights across the three datasets. The observed alignment in their trends underscores the intrinsic consistency between our method and the principles advocated in the literature. This alignment is a significant finding, as it not only validates the effectiveness of our approach but also provides empirical evidence supporting the broader implications of the Lottery Ticket Hypothesis. By demonstrating that the models trained on distinct datasets exhibit similar patterns in parameter allocation and sparsity, we reinforce the hypothesis that certain initializations, or "winning tickets," can lead to more efficient and effective training outcomes.

Table 9
The accuracy of LLaMA 7B and LLaMA2 7B models, integrated with different methods, was compared across eight commonsense reasoning datasets. The best results are in **bold**.

LLM	Method	# Params (%)	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
GPT-3_{175B}	–	–	60.5	81.0	–	78.9	70.2	68.8	51.4	57.6	–
$PaLM_{540B}$	–	–	88.0	82.3	–	83.4	81.1	76.6	53.0	53.4	–
ChatGPT	–	–	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA-7B	Prefix	0.11	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
	Series	0.99	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
	Parallel	3.54	67.9	76.4	**78.8**	69.8	78.9	73.7	57.3	75.2	72.2
	LoRA	0.41	67.8	71.9	77.9	76.3	78.5	75.7	59.5	78.0	73.2
	DoRA	0.43	**69.0**	**81.5**	78.4	70.2	80.3	78.4	63.0	77.2	74.8
	La-LoRA(ours)	0.41	68.9	81.4	77.6	**79.5**	**81.3**	**80.1**	**63.0**	**79.0**	**76.4**
LLaMA2-7B	LoRA	0.41	68.0	79.8	76.7	83.2	79.8	77.8	62.2	75.6	75.4
	DoRA	0.43	62.8	79.4	**80.1**	84.5	80.2	78.7	65.9	81.4	76.6
	La-LoRA(ours)	0.41	**71.4**	**83.0**	74.5	**84.8**	**83.3**	**84.0**	**69.8**	**81.4**	**79.0**

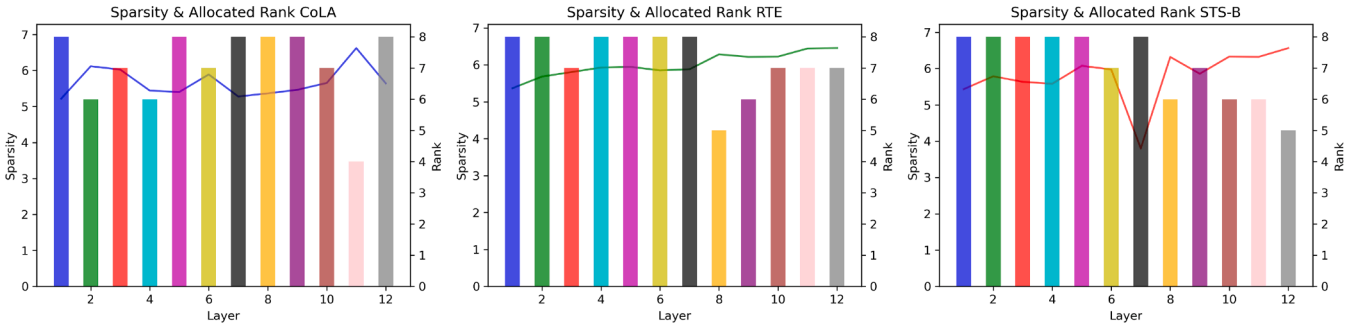


Fig. 4. The final sparsity and rank of the model are shown on the CoLA, RTE, and STS-B datasets, with the line graph representing sparsity and the bar graph representing rank.

Table 10
Performance comparison of CAR with RoBERTa base. 'With CAR' refers to gradually increasing the current total allocatable rank, while 'w/o CAR' refers to setting the current total allocatable rank to its maximum value, rL, from the beginning.

	Mathews	Accuracy	Pearson
Model	CoLA	RTE	STS-B
with CAR	**65.3**	**80.9**	**90.9**
w/o CAR	64.8	80.3	90.6

Table 11
Performance comparison of matrix relative fidelity f in Eq. (10) with RoBERTa large.

	Mathews	Accuracy	F1
Model	CoLA	RTE	MRPC
with matrix f	**69.5**	**87.0**	**92.0**
w/o matrix f	68.0	86.3	91.6

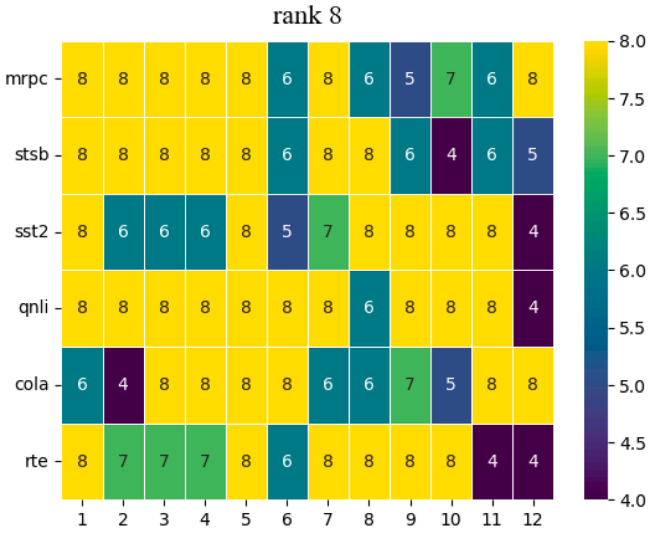
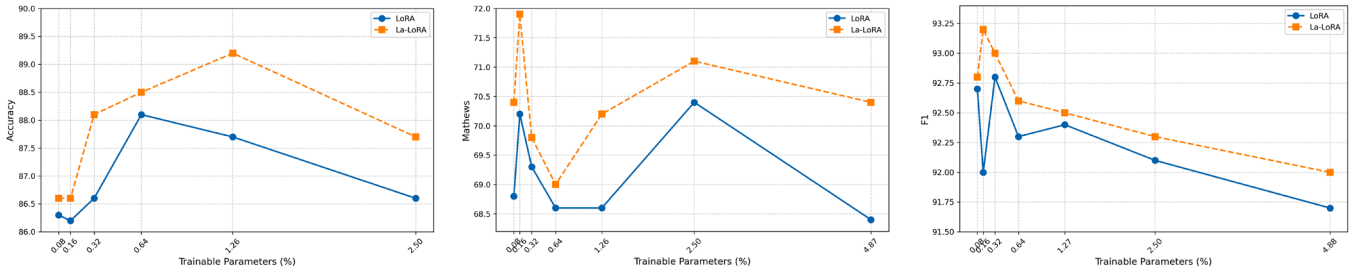


Fig. 5. The distribution of ranks per layer across different datasets with RoBERTa base.

5.5. Distribution of ranks

The rank allocation for each layer is shown in Fig. 5. As observed, La-LoRA adaptively allocates the required rank for each layer during training based on the complexity of the dataset, making full use of the limited parameter budget. From the figure, it can be observed that La-LoRA allocates higher ranks to the lower layers to capture fine-grained and more significant features. In the upper layers, La-LoRA assigns fewer ranks for simpler datasets, while for more complex datasets, it tends to allocate higher ranks to facilitate better learning by the model.

For instance, on the simpler MRPC dataset, La-LoRA reduces the parameter budget by 10.4 % compared to LoRA, while on the more complex QNLI dataset, it allocates a higher rank. This illustrates that La-LoRA, during training, regulates the growth of the total allocatable rank by monitoring the variation trend of the incremental matrix. In doing so, it effectively manages model complexity, allowing the model to adaptively adjust its complexity according to the specific demands of different tasks.



(a) RTE (b) CoLA (c) SQuADv1.1

Fig. 6. Fine-tuning performance of DeBERTa V3 base under different parameter budgets.

Table 12
Comparison of resource costs between La-LoRA and LoRA during the training stage using RoBERTa Base as the backbone model.

Dataset	Rank	Method	GPU Mem ↓	Time/epoch ↓
MNLI	8	LoRA	13.022 GB	59 min
		La-LoRA	13.020 GB	62 min
	16	LoRA	13.035 GB	59 min
		La-LoRA	13.025 GB	62 min
	32	LoRA	13.059 GB	60 min
		La-LoRA	13.037 GB	63 min
SST-2	8	LoRA	13.023 GB	10 min
		La-LoRA	13.020 GB	11 min
	16	LoRA	13.035 GB	10 min
		La-LoRA	13.025 GB	11 min
	32	LoRA	13.058 GB	10 min
		La-LoRA	13.035 GB	12 min

5.6. Resource consumption

We discuss the memory and time overheads of LoRA and La-LoRA during both the training and inference stages, using a single NVIDIA 4090D GPU.

In the training stage, Table 12 indicates that La-LoRA, compared to LoRA, does not introduce significant additional memory overhead, while it requires slightly more computational time. This slight increase in computational overhead is primarily attributed to the adaptive rank allocation mechanism used by La-LoRA during training. To optimize performance, this mechanism adjusts the rank of each layer dynamically throughout the training process, which leads to increased computational costs. However, this additional time expenditure contributes to noticeable improvements in model performance, suggesting that the mechanism strikes a balance between computational efficiency and model effectiveness.

In the inference stage, we assess the inference latency of the La-LoRA method using metrics such as Time to First Token (TTFT), Intertoken Latency (ITL), and Tokens Per Second (TPS), where each value is the average over 10 inference runs. As shown in Table 13, before merging the incremental matrix ΔW into the original weights W, both LoRA and La-LoRA exhibit a delay of approximately 3 ms in TTFT and 1 ms in ITL relative to the baseline model, attributed to the introduction of additional trainable parameters. Moreover, the adaptive rank allocation mechanism in La-LoRA results in a lower or equal rank per layer compared to LoRA, leading to a slightly reduced computational load during inference and, consequently, marginally lower inference latency. However, after the incremental matrix is merged with the original weights, no additional inference delay is observed, as the merged matrix is effectively equivalent to the original model's parameters. This results in inference performance similar to that of the original model.

Table 13
Comparison of inference latency between La-LoRA and LoRA during the inference stage, using LLaMA2-7B as the backbone model. The upper part of the table shows the latency before the incremental matrix merging, while the lower part displays the latency after the incremental matrix merging.

Method	TTFT ↓	ITL ↓	TPS ↑	GPU Mem ↓
Base	19.31 ms	18.11 ms	55 tokens/s	13.733 GB
LoRA	22.37 ms	19.26 ms	52 tokens/s	13.817 GB
La-LoRA	21.96 ms	19.15 ms	52 tokens/s	13.775 GB
Base	19.31 ms	18.11 ms	55 tokens/s	13.733 GB
LoRA	19.32 ms	18.08 ms	55 tokens/s	13.733 GB
La-LoRA	19.31 ms	18.08 ms	55 tokens/s	13.733 GB

Table 14
The effects of different contribution measurement methods on DeBERTa (above) and RoBERTa (below) are compared. '+NORM' represents the contribution measurement based on norms, while '+Fisher' represents the contribution measurement based on Fisher information.

		Accuracy	Accuracy	Mathews	EM/F1
Method		RTE	MRPC	CoLA	SQuADv1.1
LoRA		86.2	87.5	70.2	86.6/92.0
La-LoRA	+NORM	86.6	88.2	**71.9**	**87.2/93.2**
	+Fisher	**87.4**	**88.6**	71.8	87.1/93.1
LoRA		78.6	88.7	63.4	84.0/90.6
La-LoRA	+NORM	80.1	88.0	**65.0**	84.4/91.1
	+Fisher	**80.9**	**88.5**	64.8	**84.4/91.3**

5.7. Different contribution metrics

As shown in Table 14, both contribution measurement methods outperform the baseline LoRA, indicating that both methods are effective. The contribution measurement based on Fisher information performs better in complex tasks such as RTE textual entailment and SQuAD question answering, as it more accurately captures the fine-grained influence of parameters on model outputs, resulting in a 0.8% improvement for both DeBERTa and RoBERTa on the RTE task. However, for simpler tasks such as CoLA (linguistic acceptability), the norm-based method (+NORM) slightly outperforms, possibly due to the over-optimization of Fisher information on local parameter importance, leading to slight overfitting. In low-complexity tasks, the norm-based contribution measurement method demonstrates superior performance.

5.8. Different parameter budget

As shown in Fig. 6, La-LoRA consistently outperforms the baseline LoRA across various parameter budgets. The performance degradation on SQuADv1.1 with increased parameter budgets suggests that excessive model complexity may lead to mild overfitting, which conversely demonstrates the effectiveness of PEFT methods in maintaining

robust performance under constrained resources.On both RTE and CoLA datasets, models with constrained parameter budgets ($\leq 1\,\%$) achieve performance comparable to those with significantly higher parameter allocations, which may indicate the presence of trainable sparse subnetworks, in line with the Lottery Ticket Hypothesis. According to this hypothesis, critical parameter subsets, when preserved with their original initialization, could potentially deliver performance comparable to the full model.

6. Conclusion

In this paper, we introduce Layer-wise Adaptive Low-Rank Adaptation (La-LoRA). La-LoRA adaptively allocates the Dynamic Contribution-Driven Parameter Budget (DCDPB) by evaluating the contribution of each layer to the overall model. To precisely adjust the rank allocation of each layer, La-LoRA employs the Truncated Norm Weighted Dynamic Rank Allocation (TNW-DRA). The effectiveness of La-LoRA depends on its progressively increasing allocatable rank and the corrected contribution of each layer. Experimental results indicate that La-LoRA surpasses baseline methods, demonstrating that La-LoRA effectively extends LoRA and provides an effective solution for fine-tuning large-scale pre-trained models.

CRediT authorship contribution statement

Jiancheng Gu: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation; **Jiabin Yuan:** Writing – review & editing, Visualization, Validation; **Jiyuan Cai:** Writing – review & editing, Visualization, Validation; **Xianfa Zhou:** Writing – review & editing, Visualization, Validation; **Lili Fan:** Writing – review & editing, Visualization, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S. et al. (2023). Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Albert, P., Zhang, F. Z., Saratchandran, H., Rodriguez-Opazo, C., van den, H. A., & Abbasnejad, E. (2025). RandloRA: Full-rank parameter-efficient fine-tuning of large models. arXiv preprint arXiv:2502.00987.

Bi, X., Chen, D., Chen, G., Chen, S., Dai, D., Deng, C., Ding, H., Dong, K., Du, Q., Fu, Z. et al. (2024). Deepseek llm: Scaling open-source language models with longtermism. arXiv preprint arXiv:2401.02954.

Bisk, Y., Zellers, R., Le Bras, R. Gao, J. Choi, Y.(2020). Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 7432–7439). Association for the Advancement of Artificial Intelligence (AAAI) (*vol. 34*).

Bossard, L., Guillaumin, M., & Van Gool, L. (2014). Food-101–mining discriminative components with random forests. In *Computer vision–ECCV 2014: 13th European conference, Zurich, Switzerland, September 6–12, 2014, proceedings, Part VI 13* (pp. 446–461). Springer.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., M. Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Ches, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, l., Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, *33*, 1877–1901.

Cheng, G., Han, J., & Lu, X. (2017). Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, *105*(10), 1865–1883.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., V. Le, Q., Wei, J. (2024). Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, *25*(70), 1–53.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., & Toutanova, K. (2019). Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 2924–2936).

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). Think you have solved question answering? Try arc, the AI2 reasoning challenge. arXiv preprint arXiv:1803.05457.

Devlin, J. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Frankle, J., & Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635.

Gohil, V., Narayanan, S., & Jain, A. (2019). One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. arXiv preprint arXiv:1906.02773.

Guo, D., Rush, A. M., & Kim, Y. (2021). Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers)* (pp. 4884–4896).

He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T., & Neubig, G. (2021). Towards a unified view of parameter-efficient transfer learning. In *International conference on learning representations*.

He, P., Liu, X., Gao, J., & Chen, W. (2020). Deberta: Decoding-enhanced bert with disentangled attention. In *International conference on learning representations*.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). Teaching machines to read and comprehend. *Advances in Neural Information Processing Systems*, *28*, 1693–1701.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., & Gelly, S. (2019). Parameter-efficient transfer learning for NLP. In *International conference on machine learning* (pp. 2790–2799). PMLR.

Hu, E. J., shen, y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W. (2021). LoRA: Low-rank adaptation of large language models. In *International conference on learning representations*.

Kopiczko, D. J., Blankevoort, T., & Asano, Y. M. (2023). Vera: Vector-based random matrix adaptation. arXiv preprint arXiv:2310.11454.

Krizhevsky, A., Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical Report, University of Toronto.

Lateef, R. A. (2024). Machine and deep learning techniques in cancer prediction and risk stratification using bioinformatics in big data era: A review. *Edraak*, *2024*, 118–127.

Lester, B., Al-Rfou, R., & Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing* (pp. 3045–3059).

Li, X. L., & Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers)* (pp. 4582–4597).

Lialin, V., Deshpande, V., & Rumshisky, A. (2023). Scaling down to scale up: A guide to parameter-efficient fine-tuning. arXiv preprint arXiv:2303.15647.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., & Chen, M.-H. (2025). Dora: Weight-decomposed low-rank adaptation. In *Forty-first international conference on machine learning*.

Liu, Y. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

Lu, K., Grover, A., Abbeel, P., & Mordatch, I. (2022). Frozen pretrained transformers as universal computation engines. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 7628–7636). (*vol. 36*).

Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., & Zhang, Y. (2023). An empirical study of catastrophic forgetting in large language models during continual fine-tuning. arXiv preprint arXiv:2308.08747.

Mars, M. (2022). From word embeddings to pre-trained language models: A state-of-the-art walkthrough. *Applied Sciences*, *12*(17), 8805.

Meng, F., Wang, Z., & Zhang, M. (2024). Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, *37*, 121038–121072.

Mihaylov, T., Clark, P., Khot, T., & Sabharwal, A. (2018). Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 conference on Empirical methods in natural language processing* (pp. 2381–2391).

Nilsback, M.-E., & Zisserman, A. (2008). Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing* (pp. 722–729). IEEE.

Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N. A., & Kong, L. (2021). Random feature attention. arXiv preprint arXiv:2103.02143.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1* (8), 9.

Rajpurkar, P., Jia, R., & Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. arXiv preprint arXiv:1806.03822.

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 2383–2392).

Rücklé, A., Geigle, G., Glockner, M., Beck, T., Pfeiffer, J., Reimers, N., & Gurevych, I. (2021a). AdapterDrop: On the efficiency of adapters in transformers. In M.-F. Moens, X. Huang, L. Specia, & S. W.-t. Yih (Eds.), *Proceedings of the 2021 conference on Empirical methods in natural language processing* (pp. 7930–7946). Online and Punta Cana, Dominican Republic: Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.626

Rücklé, A., Geigle, G., Glockner, M., Beck, T., Pfeiffer, J., Reimers, N., & Gurevych, I. (2021b). Adapterdrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 conference on Empirical methods in natural language processing* (pp. 7930–7946).

Sakaguchi, K., Bras, R. L., Bhagavatula, C., & Choi, Y. (2021). Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, *64*(9), 99–106.

Sap, M., Rashkin, H., Chen, D., Le Bras, R., & Choi, Y. (2019). Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on Empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)* (pp. 4463–4473).

Schrimpf, M., Blank, I. A., Tuckute, G., Kauf, C., Hosseini, E. A., Kanwisher, N., Tenenbaum, J. B., & Fedorenko, E. (2021). The neural architecture of language: Integrative modeling converges on predictive processing. *Proceedings of the National Academy of Sciences*, *118* (45), e2105646118.

Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati, A., Tanzer, G., Vincent, D., Pan, Z., Wang, S. et al. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv preprint arXiv:2403.05530.

Tian, Y. (2019). Student specialization in deep reLU networks with finite width and input dimension. arXiv preprint arXiv:1909.13458.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F. et al. (2023). Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971.

Vaigandla, K. K., & Siddoju, R. K. (2025). A comprehensive review on OFDM, 5g and various PAPR minimization techniques based on machine learning. *Babylonian Journal of Networking*, *2025*, 43–58.

Valipour, M., Rezagholizadeh, M., Kobyzev, I., & Ghodsi, A. (2023). DyloRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th conference of the european chapter of the association for computational linguistics* (pp. 3274–3287).

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International conference on learning representations*.

Wang, H., Li, Y., Wang, S., Chen, G., & Chen, Y. (2025). MiloRA: Harnessing minor singular components for parameter-efficient LLM finetuning. In *Proceedings of the 2025 conference of the nations of the americas chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers)* (pp. 4823–4836).

Yu, H., Edunov, S., Tian, Y., & Morcos, A. S. (2020). Playing the lottery with rewards and multiple languages: Lottery tickets in RL and NLP. In *International conference on learning representations*.

Zaken, E. B., Goldberg, Y., & Ravfogel, S. (2022). Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 2: Short papers)* (pp. 1–9).

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., & Choi, Y. (2019). Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 4791–4800).

Zhang, L., Zhang, L., Shi, S., Chu, X., & Li, B. (2023a). LoRA-Fa: Memory-efficient low-rank adaptation for large language models fine-tuning. arXiv preprint arXiv:2308.03303.

Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., & Zhao, T. (2023b). Adaptive budget allocation for parameter-efficient fine-tuning. In *International conference on learning representations*. Openreview.