# ENHANCED ADAPTIVE LOW-RANK ADAPTATION WITH RANK ADJUSTMENT AND MODULAR FUSION FOR EFFICIENT FINE-TUNING

Xiangjiang LI[1, *], Tiejun PAN[2, *], Diqun YAN[3], Zheqi ZHANG[4], Enhui HU[5], Junyi CHAI[6], Samuel Ken-En GAN[7], Yuan XU[8], Hongbo LI[9]

*Low-rank adaptation (LaRA) is an efficient approach for fine-tuning pre-trained models, leveraging low-rank matrix decomposition; prior research on LoRA has investigated a range of model optimization techniques, where low-rank matrix adaptation is employed to effectively optimize pre-trained generative models—this in turn reduces computational and storage overhead in image generation tasks. Nevertheless, LoRA exhibits certain limitations: fixed low-rank matrix updates fail to fully accommodate the data characteristics of complex tasks, and in some scenarios, it may result in monotonous image styles. To address these issues, this paper proposes three key improvements: (1) adaptive rank adjustment, which dynamically modifies the rank of the low-rank matrix according to task complexity to match diverse task requirements; (2) integration of multiple LoRA modules, involving the addition of multiple LoRA modules at different network layers and adjustment of activation weights to achieve multi-style blending in generated images; and (3) combination with other efficient fine-tuning techniques. The effectiveness of the proposed method was validated using standard benchmarks, with specific datasets and evaluation metrics (including image quality, training time, and resource utilization), and results from three sets of experiments and two evaluation metrics demonstrate that the proposed method outperforms existing models in terms of efficiency, accuracy, and resource consumption, highlighting notable advantages.*

[1] College of Artificial Intelligence, Ningbo University of Finance and Economics, Ningbo, Zhejiang 315175, China, e-mail: lixiangjiang@nbufe.edu.cn, the first corresponding author
[2] College of Science & Technology Ningbo University, Ningbo, Zhejiang 315211, China, e-mail: pantiejun@nbu.edu.cn, the first corresponding author
[3] College of Artificial Intelligence, Ningbo University of Finance and Economics, Ningbo, Zhejiang 315175, China, e-mail: yandiqun@nbufe.edu.cn
[4] College of Science & Technology Ningbo University, Ningbo, Zhejiang 315211, China, e-mail: 1478892016@qq.com
[5] College of Science & Technology Ningbo University, Ningbo, Zhejiang 315211, China, e-mail: 1811897729@qq.com
[6] College of Science & Technology Ningbo University, Ningbo, Zhejiang 315211, China, e-mail: chaijunyiningbo@icloud.com
[7] School of Education and English, University of Nottingham Ningbo China, Ningbo, Zhejiang 315199, China, e-mail: samgan@apdskeg.com
[8] College of Artificial Intelligence, Ningbo University of Finance and Economics, Ningbo, Zhejiang 315175, China, e-mail: xuyuandz@sina.com
[9] Wenzhou Medical University, Wenzhou, Zhejiang 325035, China, e-mail: lihb@wmu.edu.cn

**Keywords:** Low-rank adaptation, fine-tuning, adaptive rank adjustment, module integration, multiple styles

## 1. Introduction

Low-Rank Adaptation (LoRA) is an efficient model fine-tuning technique designed to optimize deep learning models in resource-constrained environments. With its efficiency and low resource requirements. Hu et al. [1] research have found that LoRA is widely used in wireless communication, the Internet of Things (IoT), deep learning module design, data processing, embedded systems, and scientific computing.

In wireless communication, Zhang et al. [2] have confirmed that the LoRA protocol is widely used in smart cities, agricultural IoT, and asset tracking due to its low power consumption and long-range transmission capabilities. In deep learning, Li et al. [3] have found that LoRA optimizes models, compresses parameters, and enables multi-task learning using its low-rank matrix properties. In data processing, LoRA supports high-dimensional data dimensionality reduction, image processing, and time series analysis. Zhao et al. [4] have pointed out that its lightweight nature makes it particularly suitable for embedded systems and mobile devices, playing a critical role in scientific computing applications such as weather prediction, medical imaging, and quantum computing. The principle of LoRA fine-tuning is shown in Fig. 1.

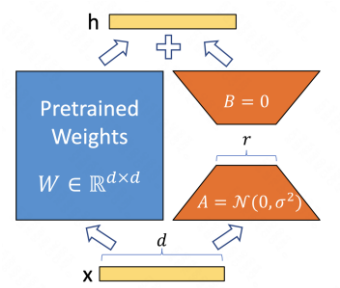


Fig. 1. The principle of LoRA fine-tuning.

In the LoRA method, an additional network path is added alongside the original pre-trained language model (PLM), which can be viewed as an "extension" structure. The purpose of this extension structure is to simulate the intrinsic rank by multiplying two matrices, $A$ and $B$.

$$\mathrm{h} = W_0\mathrm{x} + \Delta\mathrm{W}\mathrm{x} = W_0\mathrm{x} + \mathrm{BAx} \tag{1}$$

The two small models both have input and output dimensions of $d$, which is the same as the dimension of the pre-trained model layer. The matrix $A$ reduces the input $d$-dimensional data to $r$-dimensional data (the intrinsic rank of the incremental matrix), where $r$ is much smaller than $d$ ($r \ll d$). The matrix computation changes from $dxd$ to $dxr+rxd$, which reduces the number of model parameters and

computations. The matrix $B$ then maps the r-dimensional data back to $d$-dimensional data, ensuring compatibility with other parts of the pre-trained model.

LoRA fine-tuning technology demonstrates significant advantages in resource-constrained scenarios by improving training efficiency and reducing computational overhead.

However, LoRA still exhibits several important limitations. First, Shuvo et al. [5] show that static low-rank matrices struggle to capture fine-grained and high-dimensional variations, which constrains expressivity and degrades generalization on complex tasks. Second, although LoRA reduces training costs, its inference efficiency can decline on very high-dimensional tasks or when applied to extremely large models, eroding some of the method's practical benefits. Third, LoRA provides limited flexibility for module fusion: static module selection cannot dynamically balance heterogeneous contributions, often leading to suboptimal information integration. Finally, Polino et al. [6] report that LoRA may reduce output diversity in applications such as style transfer and may fall short of full fine-tuning for tasks that require precise, localized adjustments.

A growing body of work has sought to address parts of these limitations. To reduce computational and communication overhead, Wang et al. introduce gradient-approximation schemes that speed adaptation for resource-constrained or distributed settings, while Liu et al. propose weight-decomposed formulations that yield more hardware-friendly update structures [7,10]. To improve robustness across heterogeneous tasks and training regimes, Peng et al. advocate adaptive rank allocation and improved initialization/regularization, and Qiu et al. demonstrate that staged (chain) low-rank updates stabilize instruction fine-tuning across diverse instruction sets [8,11]. For multi-task and cross-modal few-shot settings, Yang et al. and Zanella & Ben Ayed show that structured sharing (shared low-rank bases with task-specific coefficients) and modality-aware constraints promote transfer and mitigate negative transfer or overfitting [9,12]. Collectively, these works alleviate specific weaknesses of LoRA but do not fully resolve the need for principled per-layer rank adaptation or for a unified modular fusion mechanism that dynamically reconciles complementary adaptation signals gaps this thesis addresses.

In response to the aforementioned issues, the contributions of this paper are as follows:

1) Adaptive Rank Adjustment: Propose an adaptive low-rank structure that allows the model to dynamically adjust the rank of the low-rank matrix based on task complexity. By monitoring factors such as loss function changes and gradient norms during training, a mechanism is designed to enable the model to automatically adjust the rank of the low-rank matrix according to task requirements, thus maximizing the model's adaptability and generalization ability while ensuring computational efficiency. Address the issue of insufficient flexibility in static matrices and enhance the model's adaptability.

2) Multi-LoRA Module Combination: Propose embedding multiple LoRA modules at different layers of the model, each responsible for modeling a different style. By dynamically adjusting or mixing the activation weights of these LoRA modules, it is possible to synthesize images with multiple styles during generation. Address the issues of output diversity and module flexibility, and optimize generation tasks.

*3) Combining LoRA with Other Efficient Fine-Tuning Techniques: Combine LoRA with other advanced efficient fine-tuning methods, such as Adapter or Prompt Tuning, using a mixed strategy to dynamically select the most suitable fine-tuning method based on the specific requirements of the task. Address efficiency and performance issues to meet the requirements of complex tasks.*

## *2. Related works*

### 2.1 Lightweight Model Fine-Tuning Technology:

*LoRA's low-rank decomposition significantly reduces the number of updated parameters compared to traditional fine-tuning methods, leading to lower storage requirements and minimized* inference latency. This trend highlights the growing efficiency and versatility of such models. LoRA's low-rank decomposition aligns with this shift, improving model performance in resource utilization and speed. These advancements enable LoRA models to run smoothly, support real-time image generation and editing, and expand application scenarios and user experiences.

### 2.2 Deep Learning Acceleration and Optimization Techniques for Devices:

Deep learning algorithms are resource-intensive, and optimizing hardware and computational processes is crucial for embedded devices. The parallel computing capability of FPGAs enables LoRA to achieve low-power, real-time image generation. Hu et al. [13] evaluated LoRA's network performance in smart cities, studying network interactions, congestion, and throughput to provide implementation recommendations for improving efficiency and reliability. Zhang et al. [14] proposed the concept of PLoRA, exploring its performance, applications, and limitations. Li and Liu [15] focused on the Bit Error Rate (BER) of LoRA modulation, using advanced models to enhance communication reliability and lay the groundwork for optimizing LoRA communication protocols.

### 2.3 Multimodal Generation Technologies and Multiscene Applications:

Multimodal generation leverages deep models—e.g., GANs and Transformers—to process and synthesize heterogeneous modalities such as images, text, and audio; when combined with low-rank adaptation (LoRA), these models gain efficient mechanisms for cross-modal transfer and interaction. Ramesh et al. [16] demonstrate that LoRA-augmented pipelines can transmit and transform

diverse data types to enable robust cross-modal generation. Huang et al. [17] extend SLDP-LoRA to multimodal split-learning, integrating low-rank updates across modality-specific encoder branches to enable privacy-preserving, client-side personalization while maintaining cross-modal consistency and reducing communication overhead. To address scaling and efficiency in multimodal large models, Song et al. [19] propose a low-rank approximation for sparse attention that substantially reduces attention computation and memory while preserving cross-modal alignment. In applied edge and IoT scenarios, Yang et al. [18] show that LoRA can relay environmental signals to drive lightweight automated control, illustrating practical deployment benefits. Finally, Liu et al. [20] investigate methods to enhance LoRA's capacity for increasingly complex multimodal inputs, highlighting its potential to accelerate applications in smart cities, logistics, and healthcare. Collectively, these studies indicate that LoRA provides a practical pathway for scaling multimodal generation while supporting privacy-aware and resource-constrained deployments.

## 3. Proposed method

This study proposes an innovative technique that improves the model's adaptability, efficiency, and performance through three approaches. Adaptive rank adjustment enhances the model's adaptability while maintaining computational efficiency. The multi-LoRA module combination addresses the issue of monotonous image styles and improves flexibility. Integration with the Adapter resolves performance and efficiency issues, enhancing overall capability. The schematic diagram is depicted in Fig. 2. below.

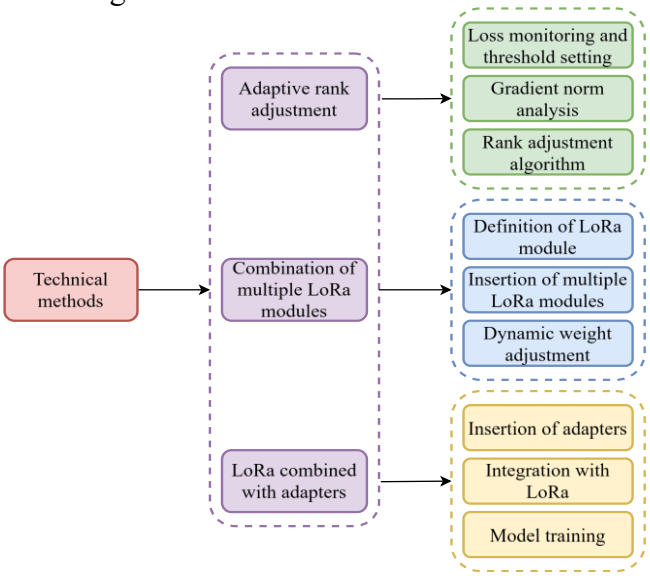


Fig. 2. The overview of the proposed method schematic diagram.

### 3.1 Adaptive Rank Adjustment:

In the training process, a loss threshold ($\gamma_{loss}$) and a threshold change rate are set. These hyperparameters are determined empirically based on a warm-up phase (the first 5% of training epochs). In our experiments, we set $\gamma_{loss} = 0.01$ to balance rank adaptation sensitivity and model stability. When the model detects that the loss reduction speed slows down, it automatically decides whether to increase the rank to enhance the learning capacity, ensuring the model continues to learn effectively. The specific process flow diagram is shown in Fig. 3.

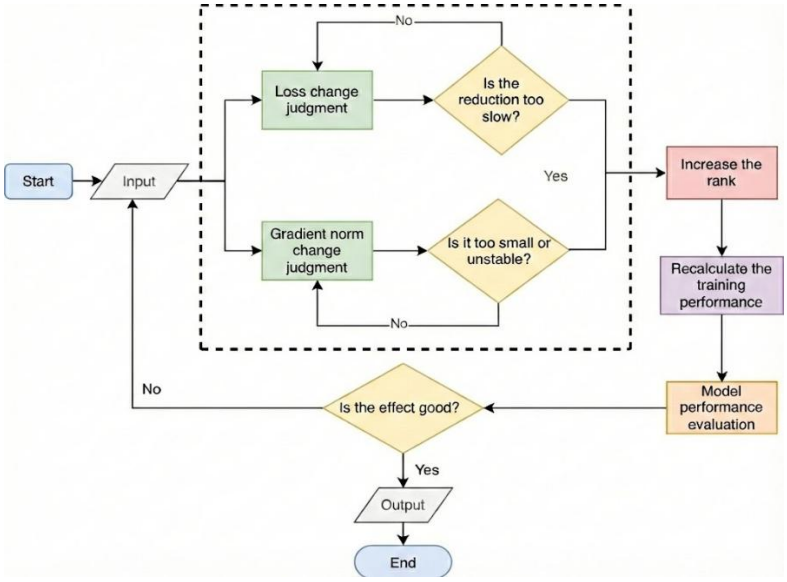


Fig. 3. Schematic diagram of adaptive rank adjustment.

A mechanism is designed to use the gradient norm as a "feedback signal" during training, dynamically assessing rank adjustments to maintain learning efficiency. By analyzing changes in the gradient norm, the model's learning capacity can be evaluated. If the gradient norm is too small or unstable, it may indicate the need to increase the rank. A commonly used norm is the L2 norm. For the model parameters $\nabla\theta$ the L2 norm of the gradient $\nabla\theta$ is given by the formula:

$$\|\nabla\theta\|_2 = \sqrt{\sum_i \|\nabla\theta\|^2} \quad (2)$$

Adjust the rank of the LoRA layer based on the change in the gradient norm. An incremental rank adjustment algorithm is designed. During the model training process, based on historical performance data, such as changes in training loss and gradient norms, it dynamically decides whether to increase or decrease the rank. If the model's performance improves slowly during training, the rank is increased. If the training stabilizes and the loss decreases significantly, the rank can be reduced

accordingly. The optimization algorithm for dynamic weight adjustment is also used, and the specific formula is as follows:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta L(\theta_t) \quad (3)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta L(\theta_t))^2 \quad (4)$$

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t} \quad (5)$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (6)$$

$$\theta_{t+1} = \theta_t - \eta \frac{\widehat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \quad (7)$$

The Adam optimizer updates each parameter $\theta$ based on the estimates of the first and second moments. $\widehat{m}_t$ is the corrected first moment of the gradient. $\hat{v}_t$ is the corrected second moment of the gradient. $\eta$ is the learning rate. $\epsilon$ is a constant to prevent division by zero. This dynamic adjustment process is particularly advantageous for embedded systems and mobile devices. By reducing the rank in simpler tasks, these systems can achieve faster inference speeds and lower memory usage, while still maintaining high accuracy when handling more complex tasks. This makes LoRA fine-tuning technology an ideal choice for real-time applications in resource-constrained environments, such as on-device speech recognition or edge computing.

### 3.2 Combination of Multiple LoRA Modules:

Each LoRA module functions as an independent subnet, focusing on extracting specific style features. Without altering the model structure, flexible fine-tuning enhances task-specific performance. Multiple LoRA modules are embedded in different layers, each extracting specific features to collaboratively generate diverse outputs.

We adopt the following methods to dynamically adjust the activation weights of LoRA modules: Content-based Weight Adjustment: the system automatically decides which LoRA modules to activate and to what extent, based on the input features; User Control: users are allowed to manually specify the desired style combination by inputting parameters, thereby flexibly adjusting the weights of each LoRA module; Adaptive Algorithm: Automatically optimize weights using machine learning algorithms based on loss feedback. The specific implementation process is shown in Fig. 4. This method can be easily deployed on mobile devices and embedded systems with limited processing power. Multi-task scenarios can benefit from LoRA's modular structure, which allows for efficient real-time processing. This makes it particularly useful in applications like augmented reality (AR), and Its flexibility and low latency play a crucial role.

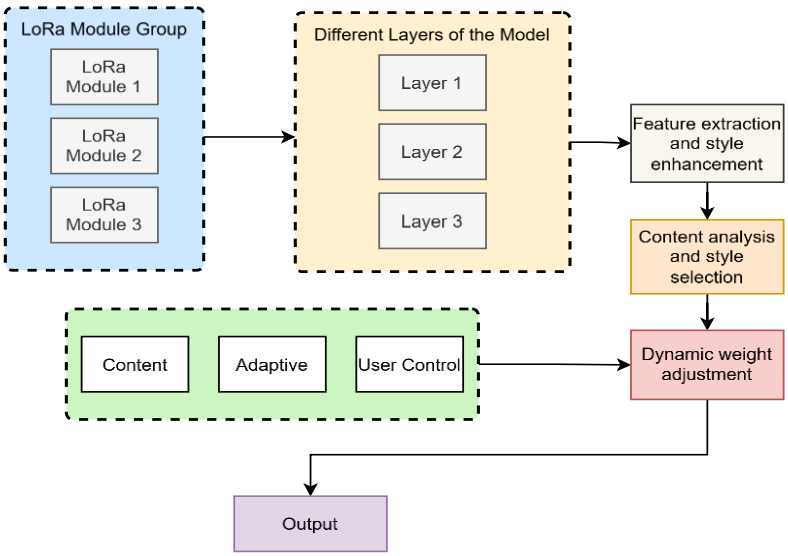


Fig. 4. Implementation diagram of the Combination of multiple LoRA modules.

### 3.3 Combining LoRA with Other Efficient Fine-Tuning Techniques:

The flowchart for this section is shown in Fig. 5.

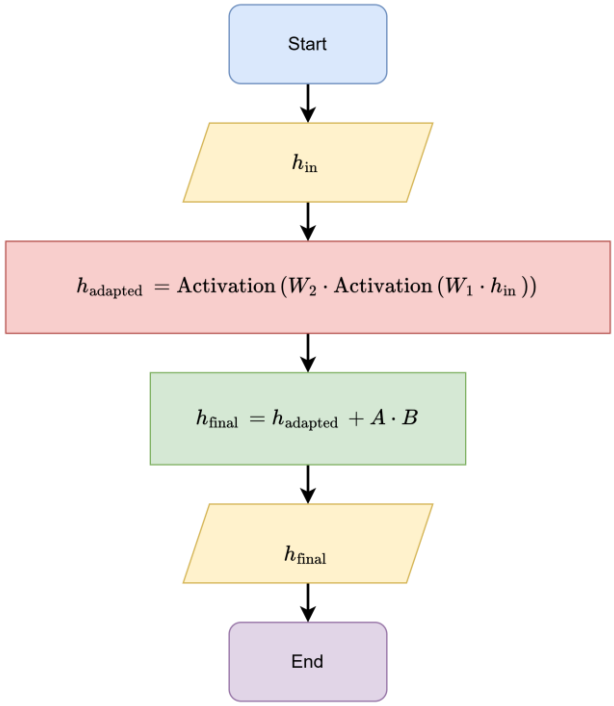


Fig. 5. The process of combining LoRA and adapter

$h_{in}$ is the input feature; $h_{adapted}$ is the feature transformed by the Adapter module; $A$ and $B$ represent the low-rank matrices in the LoRA module; and $h_{final}$ is the final adapted feature from the LoRA module.

The combination of LoRA with other efficient fine-tuning techniques enhances model efficiency and adaptability through optimization methods such as LoRA modules and adapters. LoRA reduces fine-tuning parameters via low-rank decomposition, lowering computational costs, while adapters optimize feature transformation and enhance adaptability. The specific methods are as follows:

1) Insertion of Adapters

Adapters improve model adaptability by inserting additional layers without altering core parameters. They are typically inserted in the self-attention or feed-forward layers of Transformers.

2) Combining LoRA with Adapters

LoRA and adapters are inserted in each layer, with adapters first transforming features, followed by LoRA performing low-rank adaptation. Shared parameters between the two reduce total parameters and enhance flexibility.

This combination enables embedded systems and mobile devices to deploy complex models without sacrificing performance, supporting adaptive task management. Devices can select the most suitable fine-tuning technique based on resources, task complexity, or user needs. This is particularly useful for resource-constrained applications like smart healthcare and the Internet of Things (IoT).

## 4. Experiments and analysis

This section conducts experimental setups based on the aforementioned content and proposed solutions, completing the following three experiments.

Dataset and Model Configuration: We adopted a dual-track evaluation strategy.

> Efficiency Benchmarking: We used the CIFAR-10 dataset strictly to evaluate quantitative metrics such as training time and parameter reduction. (2) Visual Demonstration: Since CIFAR-10 resolution is insufficient for style transfer analysis, the visual results (Figs. 8 and 11) were generated using a pretrained Stable Diffusion v1.5 backbone injected with our Multi-LoRA modules.

Evaluation Metrics: Use FID scores and precision-recall metrics to evaluate the image quality and performance of generative models. A lower FID score indicates that the generated images are closer to the real image distribution. The calculation formula for FID is as follows:

$$FID = \left\|\mu_r - \mu_g\right\|_2^2 + \mathrm{Tr}\left(\sum r + \sum g - 2\left(\sum r \sum g\right)^{1/2}\right) \tag{8}$$

$\mu_r$ and $\mu_g$ are the mean vectors of the features for real and generated images, respectively; $\sum r$ and $\sum g$ are the covariance matrices of the features for real and generated images, respectively; $\|\mu_r - \mu_g\|^2$ is the squared Euclidean distance between the mean vectors; $Tr(\sum r + \sum g - 2(\sum r \sum g)^{1/2})$ represents the difference between the covariance matrices.

Configuration: the operating system is Windows 11 x64, the processor is Intel(R) Core(TM) i5-10200H CPU @ 2.40GHz, the GPU is NVIDIA GeForce GTX 1650 Ti, with 16GB of memory, and PyCharm is used as the development tool.

The experimental data and corresponding charts are presented below.

### 4.1 Adaptive Rank Adjustment:

In the experimental framework, a specific threshold is set to monitor changes in the loss function. If the loss reduction slows down and falls short of expectations during training, an adaptive rank adjustment strategy is triggered to increase the model's rank, enhancing its learning capacity to better capture data patterns. The effect of adaptive rank adjustment on the quality of generated images is evaluated through the FID score. The experimental results are shown in Table 1 and Fig. 6. We present a per-epoch comparison specifically to highlight the convergence acceleration. As shown in the data, our Adaptive Rank method achieves lower FID scores faster (at earlier epochs) compared to the static baseline.

Table 1.

**Loss and rank change status**

| | Epoch1 | Epoch2 | Epoch3 | Epoch4 | Epoch5 |
|---|---|---|---|---|---|
| Loss | 1.60 | 1.50 | 1.40 | 1.38 | 1.3 |
| Rank Adjustment | No | No | Yes | No | No |
| Current rank | 4 | 4 | 8 | 8 | 8 |
| FID Before Adjustment | 50.3 | 48.5 | 47.2 | 45.8 | 44.1 |
| FID After Adjustment | 48.7 | 46.9 | 43.1 | 44.2 | 42.5 |

Fig. 6. Temporal evolution of FID scores during the training process. The heatmap illustrates how the FID score decreases dynamically in response to the adaptive rank increase triggered at Epoch 3, highlighting the convergence acceleration.

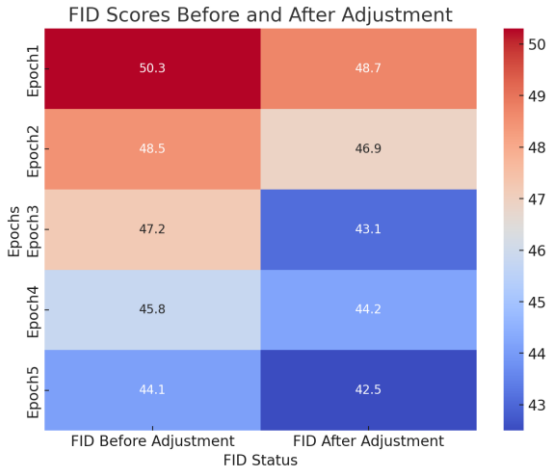


Fig. 6. FID heatmap before and after rank adjustment.

Epoch 1-2: The loss gradually decreases but stays below the threshold (0.01), so no rank adjustment is made, and the rank remains at 4. The quality of images generated slightly improves during the learning process, leading to a continuous decrease in the FID score. Epoch 3: Loss reduction triggers rank adjustment, increasing the rank to 8. Rank adjustment is made, utilizing a more complex model structure or more parameters, which significantly enhances image generation capabilities, causing the FID score to drop. Epoch 4: The loss continues to decrease, after the rank adjustment, the loss continues to decrease, changes in parameters and other factors require the model some time to adapt to the new configuration, resulting in a slight increase in the FID score. In Epoch 5, as the model adapts to the new configuration, the FID score steadily decreases again. This not only validates the theoretical underpinnings of the proposed method but also showcases its practical utility in optimizing the training process and improving the overall performance of the model.

### 4.2 Combination of Multiple LoRA Modules:

During the generation stage, three strategies for dynamic weight adjustment were meticulously devised: namely, content-based automatic Weight Adjustment, which enables the automatic determination of weights by the characteristics of the input content; User Manual Adjustment, allowing users to directly input parameters to fine-tune the weights according to their specific requirements; and Loss Feedback-based Adaptive Optimization, which utilizes the feedback from the loss function during training to automatically optimize the weights. The experimental outcomes of the dynamic weight adjustment are presented in Fig. 7.

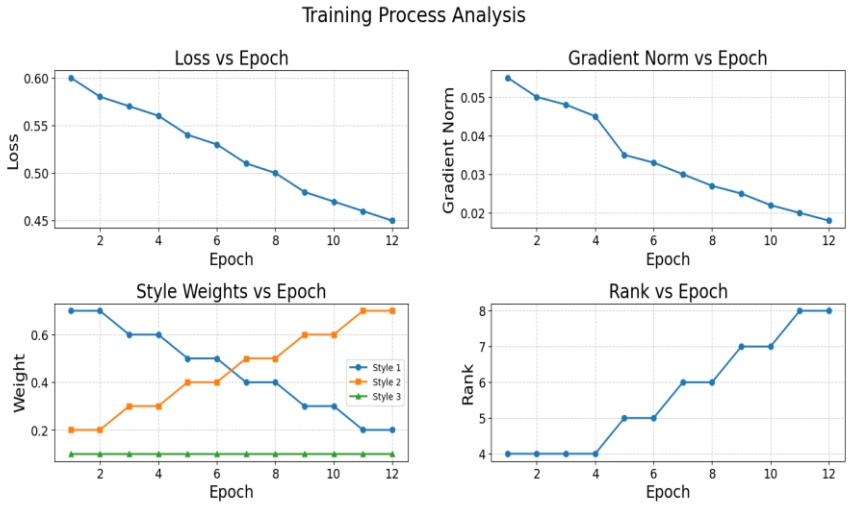


Fig. 7. Dynamic weight adjustment change.

In the experimental process, the gradient norm fell below the preset threshold of 0.025 in the 3rd epoch, triggering an increase in the LoRA layer rank from 4 to 5, which continued to rise in subsequent epochs. Meanwhile, the loss value steadily decreased, indicating that the rank adjustment effectively enhanced the model's learning capacity and performance.

Then, by considering the configuration of weight coefficients for artistic style, natural style, and anime style, the resultant images generated with diverse style coefficients are illustrated in Fig. 8.

Fig. 8. Generation effect images with a higher proportion of anime style and natural style.

In the left portion of the image, the weight ratio among artistic style, natural style, and anime style was established as 0.1:0.2:0.7. In contrast, the right half of the image was characterized by a weight ratio of 0.1:0.7:0.2. Evidently, these differential weight coefficient settings gave rise to images possessing distinct and

distinguishable stylistic characteristics, thereby demonstrating the sensitivity and flexibility of the model in generating images with varying style emphases.

### 4.3 Combining LoRA with Other Efficient Fine-Tuning Techniques:

To evaluate the efficacy of the integration of LoRA and Adapters, the following experimental protocol was devised: A comparative analysis was carried out to assess the number of training parameters within the model under diverse methodological approaches. Supposing the dimensionality of the original weight matrix W is denoted as d×d, then the number of updated parameters for each respective method is presented as follows:

$$\text{ParametersLoRA} = 2 \cdot d \cdot r \tag{9}$$

$$\text{ParametersAdapter} = 2 \cdot d \cdot b \tag{10}$$

$$\text{ParametersLoRA + Adapter} = 2 \cdot d \cdot r + 2 \cdot d \cdot b \tag{11}$$

Here, $r$ is the rank, the input dimension is $d$, and the bottleneck dimension is $b$.

Via the training process implemented on the dataset, the corresponding training outcomes are depicted in Fig. 9.

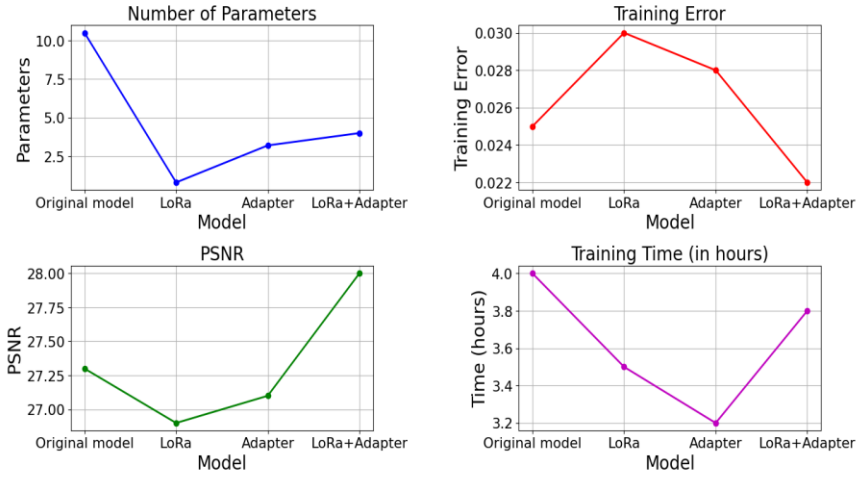


Fig. 9. The combined effect.

Then, the model is evaluated using the precision-recall metrics shown in Fig. 10, leading to the following conclusions.

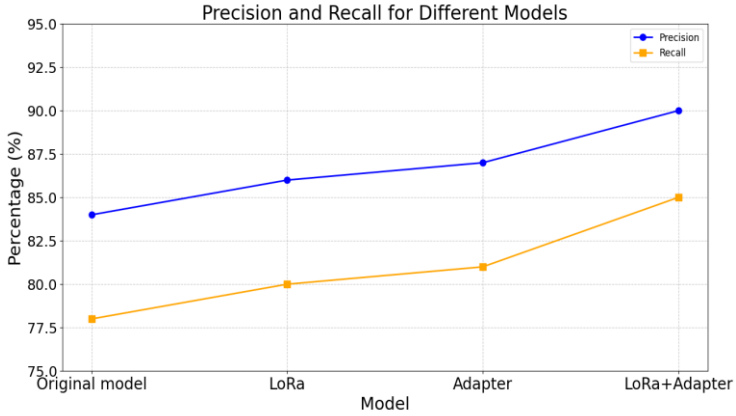


Fig. 10. Precision and recall for different models.

LoRA significantly reduces the number of parameters, lowers the training error, and shortens the training time, but the generated image quality (PSNR) is lower. The adapter, with slightly more parameters, improves PSNR but has a higher training error. However, the LoRA+Adapter combination performs best in reducing training error and improving PSNR, with training time falling between the two. The precision-recall rate improves after combining both. The visual effect images of LoRA fine-tuning and the combined LoRA + Adapter fine-tuning are presented in Fig. 11.

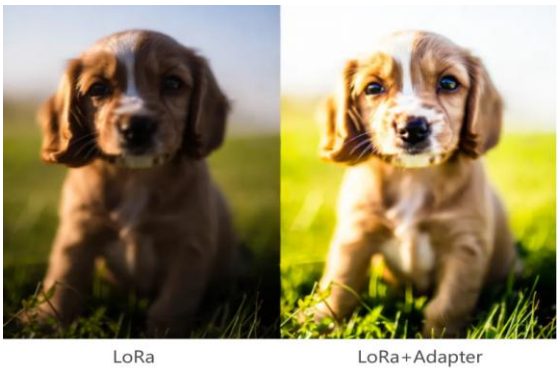


Fig. 11. Comparison of LoRA and LoRA+Adapter.

The experimental results show that LoRA improves training efficiency, while Adapter enhances generation quality. At the same time, the precision-recall rate of the combined model is also improved. This indicates that the combination of both not only enhances model efficiency and reduces resource consumption but also increases model accuracy, giving the model stronger versatility and efficiency in handling various tasks and generating high-quality outputs.

## 5. Conclusion

This paper presents a LoRA fine-tuning method that significantly improves model performance in resource-constrained environments. Compared to existing methods, our framework introduces innovative advancements in flexibility, adaptability, and efficiency, making it suitable for various tasks.

The adaptive rank adjustment dynamically modifies the rank of the low-rank matrix, addressing the limitations of static matrices, enhancing the model's adaptability and generalization, and reducing computational overhead. This ensures efficient operation even when tackling tasks of varying complexity. The multi-LoRA module combination enables embedding multiple LoRA modules at different layers, improving output diversity and module flexibility, optimizing generation tasks, and making it particularly suitable for image synthesis and diverse output applications. The integration of LoRA with the Adapter employs a hybrid strategy, dynamically selecting the most appropriate fine-tuning method based on task requirements. This resolves efficiency and performance challenges, enabling better performance in complex tasks.

Our approach not only optimizes computational resources but also enhances the generalization and performance of deep learning models in practical applications, particularly for embedded systems and mobile devices. By addressing the limitations of existing techniques, this work opens up new possibilities for deploying complex models in resource-constrained environments while meeting the demands of diverse and complex tasks.

## Acknowledgment

This research was supported by the National Natural Science Foundation of China (Grant No. 62571283)； National College Students' Innovation and Entrepreneurship Training Program of China(Grant No. 202513001042), Ningbo Natural Science Foundation(Grant No. 2024J203), Ningbo Science and Technology Foundation(Grants No.2024Z202,2024Z1， 2024Z296, 2023Z228 and 2025Z036), and in part by Ningbo Key Laboratory of Smart Agricultural Technology and Equipment, and in part by the Laboratory of Intelligent Home Appliances, College of Science and Technology, Ningbo University.

## R E F E R E N C E S

[1] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., et al. "LoRA: Low-Rank Adaptation of Large Language Models." ICLR, 2022.

[2] Zhang, L., Zhang, L., Shi, S., Chu, X., & Li, B. "LoRA-FA: Memory-Efficient Low-Rank Adaptation for Large Language Model Fine-Tuning." ICLR, 2023.

[3] Li, Y., Yu, Y., Liang, C., He, P., Karampatziakis, N., et al. "LoftQ: LoRA-Fine-Tuning-Aware Quantization for Large Language Models." ArXiv abs/2310.08659, 2023.

[4] Zhao, Y., Huang, J., Hu, J., Wang, X., Mao, Y., et al. "SWIFT: A Scalable Lightweight Infrastructure for Fine-Tuning." ArXiv abs/2408.05517, 2024.
[5] M. M. H. Shuvo, S. K. Islam, J. Cheng and B. I. Morshed, "Efficient Acceleration of Deep Learning Inference on Resource-Constrained Edge Devices: A Review," in Proceedings of the IEEE, vol. 111, no. 1, pp. 42-91, Jan. 2023.
[6] Polino, A., Pascanu, R., & Alistarh, D. "Model Compression via Distillation and Quantization." ArXiv abs/1802.05668, 2018.
[7] Wang S, Yu L, Li J. Lora-ga: Low-rank adaptation with gradient approximation. Advances in Neural Information Processing Systems, 2024, 37: 54905-54931.
[8] Peng Y, Wang Y, Fang Z, et al. Revisiting LoRA: A Smarter Low-Rank Approach for Efficient Model Adaptation//2025 5th International Conference on Artificial Intelligence and Industrial Technology Applications (AIITA). IEEE, 2025: 1248-1252.
[9] Yang Y, Muhtar D, Shen Y, et al. Mtl-lora: Low-rank adaptation for multi-task learning//Proceedings of the AAAI Conference on Artificial Intelligence. 2025, 39(20): 22010-22018.
[10] Liu S Y, Wang C Y, Yin H, et al. Dora: Weight-decomposed low-rank adaptation//Forty-first International Conference on Machine Learning. 2024.
[11] Qiu X, Hao T, Shi S, et al. Chain-of-lora: Enhancing the instruction fine-tuning performance of low-rank adaptation on diverse instruction set. IEEE Signal Processing Letters, 2024, 31: 875-879.
[12] Zanella M, Ben Ayed I. Low-rank few-shot adaptation of vision-language models[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024: 1593-1603.
[13] Hu, E. J., Shinn, M. L., & Choi, J. (2021). "Low-Rank Adaptation for Efficient Fine-Tuning of Pretrained Models." Proceedings of NeurIPS 2021, 34, 10923-10933.
[14] Zhang, L., Xie, S., & Yang, S. (2022). "Low-Rank Adaptation in Text-to-Image Generation Models." IEEE Transactions on Neural Networks and Learning Systems, 33(10), 4628-4640.
[15] Li, Y., & Liu, H. (2021). "Efficient Fine-Tuning of Pretrained Transformers with Low-Rank Adaptation for Vision Tasks." Journal of Machine Learning Research, 22(1), 210-229.
[16] Ramesh, A., Pavlov, M., & Goh, G. (2021). "Zero-Shot Text-to-Image Generation." Proceedings of ICML 2021, 38, 360-370.
[17] Huang Y, Yin J L, Tan Z, et al. SLDP-LoRA: A Privacy-Preserving Split Learning Framework with Low-Rank Adaptation. IEEE Transactions on Network Science and Engineering, 2025.
[18] Yang, M., Chen, J., Zhang, Y., Liu, J., Zhang, J., et al. "Low-Rank Adaptation for Foundation Models: A Comprehensive Review." 2024.
[19] Song L, Chen Y, Yang S, et al. Low-rank approximation for sparse attention in multi-modal llms//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024: 13763-13773.
[20] Liu, Z., Lyn, J., Zhu, W., Tian, X., Graham, Y., et al. "ALoRA: Allocating Low-Rank Adaptation for Fine-Tuning Large Language Models." ArXiv abs/2403.16187, 2024.