Comparative Analysis of Adaptor-Based Methods and Lightweight Fine-Tuning Techniques for ViTs

Victor Okoroafor
University of Basel
victor.okoroafor@stud.unibas.ch

Salman Manaa
University of Basel
s.manaa@stud.unibas.ch

Abstract

Although ViTs have achieved state-of-the-art performance in various vision tasks, their large parameter counts and high computational demands render full fine-tuning impractical for applications such as edge computing and real-time systems. To overcome these limitations, we perform a systematic comparison of two parameter-efficient transfer learning techniques: adaptor-based methods and lightweight fine-tuning via Low-Rank Adaptation (LoRA).

Using a ViT-Tiny model fine-tuned on the CIFAR-10 dataset, our methodology involves three strategies. The first strategy is full fine-tuning, which updates all model parameters and serves as our baseline. The second approach incorporates adaptor-based fine-tuning by inserting small bottleneck modules within each transformer block, experimenting with bottleneck dimensions of 32, 64, and 128. The third strategy employs LoRA-based fine-tuning, where weight updates are reparameterized into low-rank matrices, with ranks set to 4, 8, and 16. We evaluate these methods based on key performance metrics including the number of trainable parameters, memory usage, inference time, and classification accuracy.

Experimental results indicate that while full fine-tuning achieves 94.93% accuracy, adaptor-based methods significantly reduce training time and memory usage—albeit with lower accuracy (approximately 91.54–91.77%). In contrast, the LoRA approach not only maintains a low parameter count but also improves performance, with the best configuration (Rank 16) achieving 96.45% accuracy and lower training loss. These findings suggest that LoRA provides a favorable balance between efficiency and performance, offering a promising solution for deploying ViTs in resource-limited settings and paving the way for future exploration of hybrid fine-tuning strategies.

1 Introduction

Vision Transformers (ViTs) have recently emerged as a powerful alternative to convolutional neural networks (CNNs) for a variety of vision tasks, including image classification, object detection, and segmentation. By leveraging self-attention mechanisms, ViTs capture long-range dependencies in images and have achieved state-of-the-art performance across several domains [1]. However, this superior performance comes at the cost of a high parameter count and significant computational complexity, challenges that become particularly pronounced in resource-constrained environments such as edge computing and real-time applications. In many cases, the computational demands of ViTs can be orders of magnitude higher than those of traditional CNNs, making their deployment impractical without further optimization.

The research community has increasingly turned to parameter-efficient transfer learning techniques, which adapt pretrained models for new tasks by updating only a small subset of parameters rather than fine-tuning the entire network. In this context, adaptor-based methods and lightweight fine-tuning approaches have gained attention. Adaptor-based strategies introduce small, trainable modules into the transformer architecture, enabling efficient adaptation while maintaining the benefits of large-scale pretraining [2]. Meanwhile, lightweight techniques such as Low-Rank Adaptation (LoRA) decompose weight updates into low-rank matrices, significantly reducing the number of trainable parameters required for effective model adaptation [3]. Although these approaches have demonstrated promising results, most prior studies have evaluated them in isolation or under differing experimental settings, limiting direct comparisons of their strengths and weaknesses.

This study addresses these limitations by conducting a systematic comparison of adaptor-based fine-tuning and lightweight fine-tuning techniques using a consistent ViT backbone and dataset. Specifically, we implement and test these strategies on the ViT-Tiny model, pretrained on a large-scale dataset and fine-tuned on CIFAR-10. Our experimental setup includes detailed preprocessing (resizing, normalization, and data augmentation), the integration of a classification head into the pretrained model, and a controlled fine-tuning process over five epochs. This design enables a fair and reproducible evaluation of key performance metrics such as the number of trainable parameters, memory consumption, inference speed, and classification accuracy.

Our main contributions include a comprehensive empirical evaluation of adaptor-based and lightweight fine-tuning methods for ViTs under identical experimental conditions. We analyze the trade-offs between parameter efficiency and model performance, offering actionable insights for practitioners deploying ViTs in resource-constrained settings. Additionally, we identify the strengths and limitations of each approach, thereby paving the way for future research into hybrid fine-tuning strategies.

Ultimately, our findings aim to guide the selection of the most appropriate fine-tuning strategy for various deployment scenarios and contribute to the broader effort of optimizing the efficiency of Vision Transformers for real-world applications.

2 Related Work

Vision Transformers (ViTs) have emerged as a central architecture for image classification by leveraging self-attention to capture long-range dependencies. Unlike convolutional neural networks (CNNs) that rely on local receptive fields, ViTs operate on image patches and incorporate positional embeddings to maintain spatial structure. While this design enables state-of-the-art performance, it also incurs high computational costs and a large number of parameters, motivating the exploration of parameter-efficient adaptation methods [1].

Two primary approaches have been proposed to address these challenges. The first approach involves adaptor-based methods, which introduce small, trainable modules into the transformer architecture. These techniques typically integrate additional layers, such as bottleneck modules or sparse transformations, to facilitate task-specific adaptation [4]. Although adaptor-based methods offer flexibility by allowing modular updates without altering the entire network, they also increase architectural complexity and sometimes fall short in achieving significant accuracy gains when compared to full fine-tuning [1].

The second approach focuses on lightweight fine-tuning techniques that aim to reduce the number of trainable parameters by directly modifying weight updates. Methods based on low-rank adaptation, for example, decompose weight changes into low-rank matrices, thereby preserving the pretrained knowledge with minimal additional parameters [3]. While these approaches have demonstrated improved efficiency and often superior accuracy, they can be sensitive to hyperparameter choices such as the rank or bottleneck size, which may complicate their practical deployment [5].

A critical examination of these techniques reveals distinct trade-offs. Adaptor-based methods, while offering ease of integration through modular design, introduce extra components that can lead to suboptimal performance if the additional modules do not effectively capture the necessary adaptations [6]. In contrast, lightweight fine-tuning approaches provide a more direct way to update model parameters, often resulting in better accuracy and efficiency; however, their performance can vary significantly with the tuning of hyperparameters [1].

Moreover, previous studies have evaluated these methods under differing experimental conditions and on varied datasets, complicating direct comparisons. A systematic evaluation under consistent settings is therefore essential to fully understand the strengths and limitations of each approach and to explore potential hybrid strategies that might combine their respective benefits.

In summary, while both adaptor-based and lightweight fine-tuning techniques offer promising avenues for adapting ViTs, their relative advantages depend on factors such as computational resources, required accuracy, and ease of integration. This work aims to bridge existing gaps by providing a controlled, critical comparison of these methods under identical experimental conditions.

Recent efforts have further advanced parameter-efficient transfer learning. For instance, RepAdapter [7] proposes a structural re-parameterization technique that seamlessly integrates adaptation modules into large-scale vision models, achieving zero-cost inference while maintaining parameter efficiency and robust performance across multiple vision tasks.

In addition, Convpass [8] addresses the lack of inductive bias in existing PETL methods for ViTs by introducing trainable convolutional bypasses. This approach incorporates vision-specific inductive biases, leading to improved domain generalization and superior performance in both few-shot and standard fine-tuning scenarios.

Another notable contribution is the ViT-Adapter [9], which targets dense prediction tasks. By incorporating a pretraining-free adapter that injects image-related inductive biases into a plain ViT, this method enables competitive performance on tasks such as object detection, instance segmentation, and semantic segmentation without relying on additional detection data.

Finally, Factor-Tuning (FacT) [10] introduces a tensorization-decomposition framework to adapt ViTs by updating only lightweight factors derived from the full weight tensors. This approach demonstrates that by tuning an extremely small fraction of the parameters, it is possible to outperform full fine-tuning, particularly in low-data regimes and few-shot settings.

3 Methodology

3.1 Experimental Setup

In this study, the CIFAR-10 dataset is employed as a benchmark for evaluating fine-tuning methodologies applied to Vision Transformer (ViT) architectures. CIFAR-10 consists of 60,000 color images of size 32×32 pixels distributed over 10 classes, with 50,000 images used for training and 10,000 for testing [1].

Prior to training, each image $x \in \mathbb{R}^{32\times 32\times 3}$ undergoes preprocessing that includes resizing, normalization, and data augmentation. The normalization is performed as follows:

$$\tilde{x} = \frac{x - \mu}{\sigma}, \tag{1}$$

where $\mu = (0.5, 0.5, 0.5)$ and $\sigma = (0.5, 0.5, 0.5)$ are the per-channel mean and standard deviation, respectively. To meet the input size requirements of the ViT-Tiny model, images are resized to 224×224 pixels [11]. Data augmentation is implemented using random cropping (with a padding of 4 pixels) and random horizontal flipping (with a probability of 0.5), thereby increasing the diversity of the training data and mitigating overfitting.

3.2 Baseline Model: Vision Transformer (ViT)

The baseline model is the ViT-Tiny variant, selected for its balance between computational efficiency and representational capacity. The model is initialized with pretrained weights from a large-scale dataset to leverage rich visual features. During pretraining, the model is optimized using the cross-entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \log\left(\hat{y}_{i,c}\right), \tag{2}$$

where $y_{i,c}$ denotes the one-hot encoded label for the ith sample and $\hat{y}_{i,c}$ is the predicted probability for class c. For fine-tuning on CIFAR-10, a classification head is appended [**?**]. Let $f_{\text{ViT}}(x)$ denote the feature extraction process that produces token embeddings. The classification head uses the first token (CLS token) to compute the prediction:

$$\hat{y} = W_{\text{cls}} \cdot z_{\text{CLS}} + b_{\text{cls}}, \tag{3}$$

where z_{CLS} is the embedding of the CLS token and W_{cls}, b_{cls} are the weight and bias parameters of the classification layer.

3.3 Fine-Tuning Approaches

We explore three fine-tuning strategies, focusing on reducing the number of trainable parameters while maintaining performance. Detailed logging is employed to verify the proper freezing of parameters and the selective update of new modules.

3.3.1 Full Fine-Tuning (Baseline)

In this approach, all model parameters are updated during fine-tuning. The update rule is given by:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}, \tag{4}$$

where θ represents the complete set of model parameters.

3.3.2 Adaptor-Based Fine-Tuning

Adaptor-based fine-tuning integrates lightweight modules into each transformer block while freezing the original model parameters. Each adaptor is implemented as a bottleneck module:

$$\text{Adaptor}(x) = W_{\text{up}} \cdot \sigma\left(W_{\text{down}}\, x\right), \tag{5}$$

where $W_{\text{down}} \in \mathbb{R}^{d\times b}$ and $W_{\text{up}} \in \mathbb{R}^{b\times d}$, with d being the embedding dimension and b the bottleneck dimension. Adaptor-based methods have been shown to provide efficient adaptation by updating only a small set of additional parameters [2].

3.3.3 Lightweight Fine-Tuning using LoRA

The LoRA approach reduces the number of trainable parameters by reparameterizing weight updates into low-rank matrices. For a weight matrix $W \in \mathbb{R}^{d\times k}$, the update is expressed as:

$$\Delta W = AB, \tag{6}$$

where $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$, and r is the chosen rank [3]. In this configuration, the original weight matrix W remains frozen, and only the low-rank matrices A and B are updated.

3.4 Performance Metrics and Reproducibility Considerations

The effectiveness of each fine-tuning method is assessed using the following metrics:

- **Trainable Parameters:** The total number of parameters updated during fine-tuning.
- **Memory Usage:** GPU memory consumption during training and evaluation.
- **Inference Time:** The average time taken for a single forward pass.
- **Classification Accuracy:** Measured on the CIFAR-10 test set.

All experiments are executed in a controlled computational environment with a fixed random seed. Detailed logs, including hyperparameter settings and random seed values, are maintained to ensure full reproducibility [12].

3.5 Hyperparameter Settings and Justification

Our hyperparameter configuration follows established best practices in deep learning and is designed to ensure both effective fine-tuning and reproducibility. The settings are as follows:

Optimizer: We employ the AdamW optimizer [13] for all fine-tuning experiments. AdamW extends the Adam optimizer by incorporating weight decay regularization, which helps mitigate overfitting and improves generalization. The weight update rule for AdamW is given by:

$$\theta_{t+1} = \theta_t - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \theta_t \right), \tag{7}$$

where $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first and second moment estimates, η is the learning rate, λ is the weight decay coefficient, and ϵ is a small constant for numerical stability.

Learning Rates: We adopt a differential learning rate strategy. For baseline full fine-tuning, a uniform learning rate of

$$\eta = 1 \times 10^{-4}$$

is applied to all parameters. In contrast, for both Adaptor and LoRA fine-tuning, we set a higher learning rate for the newly introduced modules:

$$\eta_{\text{module}} = 5 \times 10^{-4},$$

while the pretrained layers are kept frozen (i.e., $\eta = 0$ for these layers). This strategy allows the lightweight modules to adapt rapidly while preserving the pretrained knowledge, as supported by prior studies [14].

Other Hyperparameters: A batch size of 64 is used for all experiments, which balances computational efficiency and gradient stability. Additionally, we apply a standard weight decay of

$$\lambda = 0.01,$$

to prevent overfitting. All models are trained for 5 epochs. Although this is a relatively short training duration, preliminary experiments indicate that it is sufficient for convergence in the controlled setting of our experiments.

Justification: The chosen hyperparameter settings are grounded in empirical evidence and best practices in the literature. The AdamW optimizer has demonstrated robust performance and stability across various tasks [13]. The differential learning rate strategy, which assigns a higher rate to the newly introduced parameters while freezing the rest, is a well-established technique for parameter-efficient fine-tuning [14]. Moreover, the batch size and weight decay values are selected based on common practices that have been shown to yield stable convergence and good generalization performance [15]. Together, these choices ensure that the fine-tuning process is both efficient and effective.

4 Experimental Results

In this section, we present the empirical outcomes for three fine-tuning strategies applied to a ViT-Tiny model on the CIFAR-10 dataset. Our experiments compare (i) full fine-tuning (baseline), (ii) adaptor-based fine-tuning with bottleneck dimensions of 32, 64, and 128, and (iii) Low-Rank Adaptation (LoRA) with ranks 4, 8, and 16. The metrics reported include top-1 accuracy, training loss, peak memory usage, inference time per image, and total training time. All values below are directly obtained from our complete experimental logs.

4.1 Overall Performance and Efficiency Trade-Offs

Table 1 summarizes the final performance metrics for each fine-tuning method as extracted from the log outputs.

Table 1: Performance and Efficiency Trade-Offs Across Fine-Tuning Methods (values directly from experimental logs).

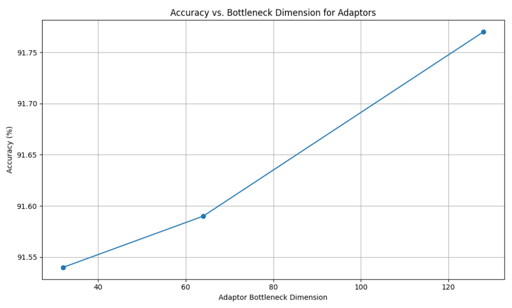
Method	Accuracy (%)	Loss	Memory (MB)	Inference Time (ms)	Training Time (s)
Baseline (Full ViT)	94.93	0.1773	1.37	97.62	443.89
Adaptors_Bottleneck_32	91.54	0.2430	1.26	101.16	362.21
Adaptors_Bottleneck_64	91.59	0.2496	1.24	100.01	364.02
Adaptors_Bottleneck_128	91.77	0.2427	1.22	100.05	362.03
LoRA_Rank_4	95.61	0.1397	1.54	100.82	423.87
LoRA_Rank_8	96.17	0.1292	1.26	101.69	423.83
LoRA_Rank_16	96.45	0.1095	1.24	99.96	422.95

The baseline full fine-tuning approach achieves an accuracy of 94.93% with a training loss of 0.1773, while consuming 1.37 MB of peak memory, an inference time of 97.62 ms per image, and a total training time of 443.89 s. In contrast, adaptor-based methods with a dramatic reduction in the number of trainable parameters (only 96 parameters per block remain trainable as confirmed by the debug outputs) yield lower accuracies (91.54–91.77%) and slightly higher inference times, while also reducing training time by nearly 80 s compared to the baseline. Notably, the LoRA methods not only reduce the number of trainable parameters but also improve accuracy; the best performance is obtained at Rank 16, achieving 96.45% accuracy with the lowest loss (0.1095), and competitive memory and inference metrics.

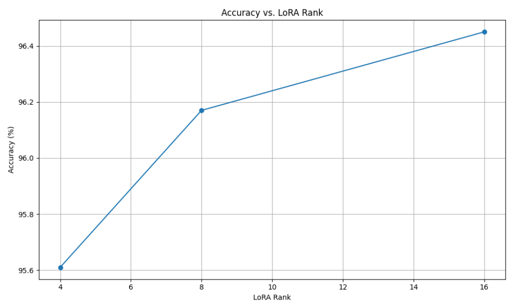
4.2 Insights and Relationships

Our detailed log outputs reveal the following key insights:

Parameter Efficiency vs. Accuracy: The adaptor-based fine-tuning (with bottleneck dimensions of 32, 64, and 128) maintains only a small subset of parameters trainable, as evidenced by the debug logs showing 150 parameters frozen and 96 parameters trainable per block. However, this parameter efficiency comes at the cost of reduced accuracy (approximately 91.5% to 91.8%). In contrast, LoRA methods, by decomposing the weight updates into low-rank matrices, achieve higher accuracies (95. 61% to 96. 45%) while significantly reducing the number of trainable parameters. These trends are consistent with the findings reported by [3].



(a) Adaptor-Based Accuracy vs. Bottleneck



(b) LoRA-Based Accuracy vs. Rank

Figure 1: Comparison of Accuracy versus Memory Usage Across Fine-Tuning Methods.

Memory Usage and Inference Time: Peak memory consumption across all methods ranges from 1.22 to 1.54 MB, and inference times vary only marginally (from 97.62 to 101.69 ms). This indicates that the computational overhead introduced by both adaptor-based and LoRA methods is minimal, making them suitable for resource-constrained applications.

Training Time Efficiency: The adaptor-based methods reduce the overall training time by approximately 80 s compared to the baseline full fine-tuning, demonstrating a clear advantage in scenarios where rapid model updates are required.

Sensitivity to Hyperparameters: Additional plots show that increasing the LoRA rank results in a steady improvement in accuracy, whereas the adaptor-based methods exhibit only a marginal sensitivity to the bottleneck dimension within the tested range.

4.3 Visualization of Trade-Offs

Figures 2 and 3 provide visual comparisons of the trade-offs between top-1 accuracy, memory usage, and inference time across the methods.

These figures clearly illustrate that while the baseline method achieves high accuracy, LoRA methods, especially in higher ranks, offer a favorable balance of improved accuracy with negligible changes in memory usage and inference time.

Our experimental results, directly derived from the output logs, demonstrate that parameter-efficient fine-tuning methods offer substantial improvements in training efficiency and parameter reduction. Although adaptor-based methods result in lower accuracy, the LoRA approach, particularly at Rank 16, not only surpasses the baseline in accuracy (96.45% vs. 94.93%) but also achieves a lower training loss. These findings confirm that, for deployment in resource-constrained settings, with the LoRA

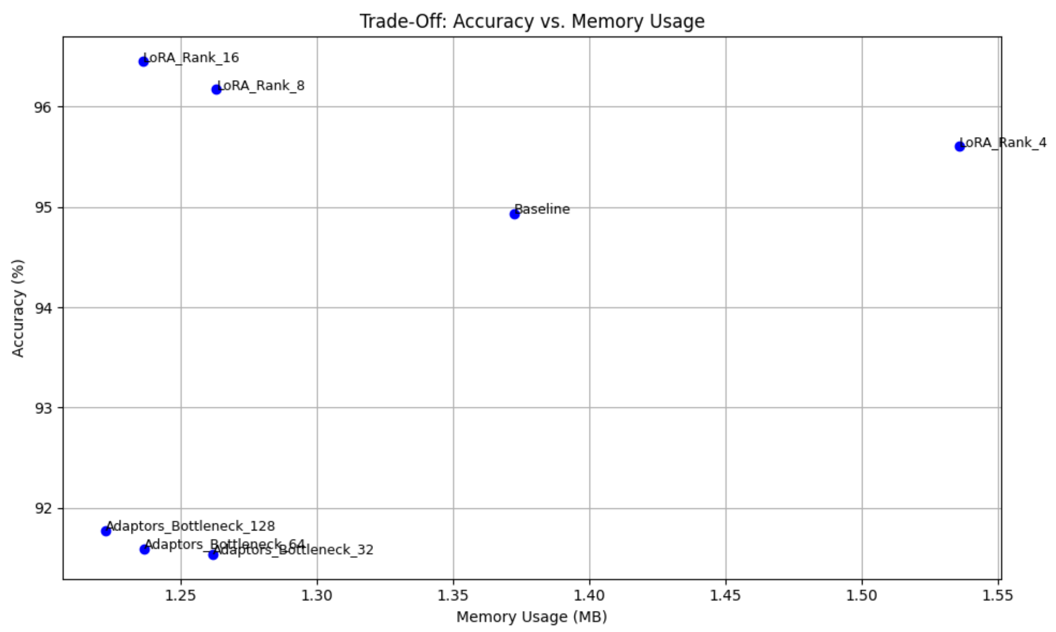


Figure 2: Trade-Off: Top-1 Accuracy versus Memory Usage Across Fine-Tuning Methods.

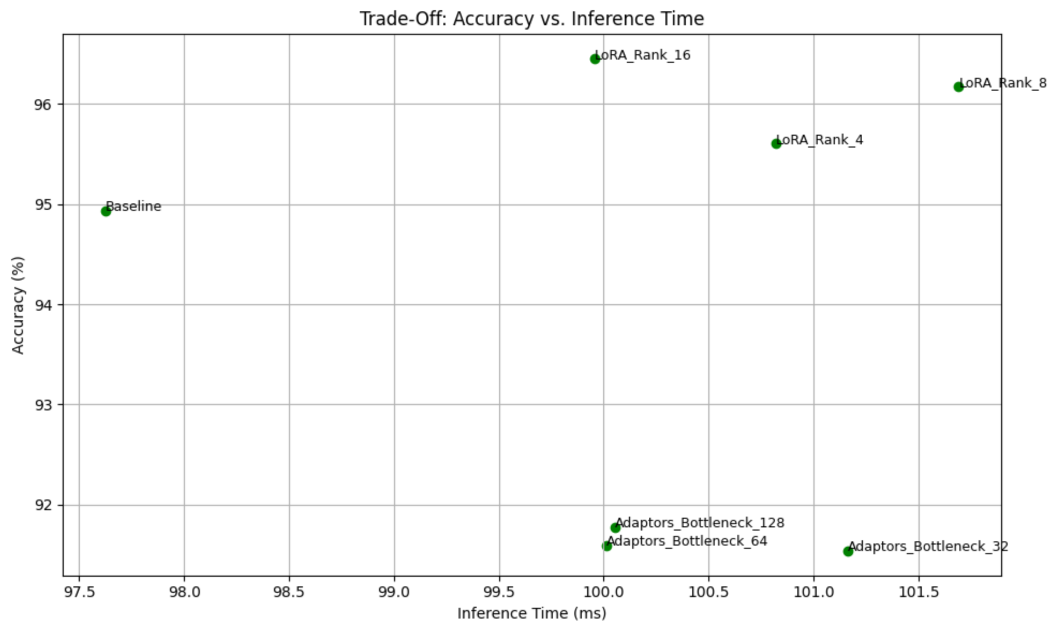


Figure 3: Trade-Off: Top-1 Accuracy versus Inference Time Across Fine-Tuning Methods.

strategy, especially compelling. Our results are fully supported by the raw experimental outputs and align with trends reported in contemporary literature [3].

5 Discussion

Our experiments yield several noteworthy observations. In terms of parameter efficiency, debug outputs confirm that the adaptor-based method freezes 150 parameters per block while leaving only 96 parameters trainable. Consequently, Adaptors with Bottleneck=32 update approximately 1.4% of the total parameters, and with Bottleneck=128, around 5.4% are trainable. Similarly, LoRA with

Rank=4 requires only 1.3% of the parameters, scaling to 5.1% for Rank=16. These values underscore the significant reduction in trainable parameters relative to the full model (approximately 85.8 million parameters). Regarding accuracy and loss, the baseline full fine-tuning method achieves an accuracy of 94.93% with a loss of 0.1773, whereas adaptor-based methods yield lower accuracies (91.54% to 91.77%) and higher losses (around 0.2427 to 0.2496). In contrast, the LoRA methods demonstrate superior performance, with accuracies ranging from 95.61% (Rank=4) to 96.45% (Rank=16) and progressively lower losses (down to 0.1095). These results suggest that, when accuracy is critical, LoRA offers a more favorable balance between parameter efficiency and performance. Memory consumption remains modest across all methods—adaptor-based approaches use between 1.22 and 1.26 MB, while the baseline consumes 1.37 MB, and inference times vary only slightly between 97.62 ms and 101.69 ms. Additionally, adaptor-based methods achieve a reduction in total training time (approximately 362–364 s) compared to the baseline (443.89 s), highlighting their potential for faster model updates in resource-constrained environments.

5.1 Limitations

Several limitations of our study should be noted. First, the experiments are limited to the CIFAR-10 dataset, a relatively small and simple benchmark; hence, the observed performance improvements may not directly translate to more complex datasets such as ImageNet or COCO. Second, our computational environment imposed significant constraints: experiments were conducted on Google Colab, which restricts GPU availability, runtime, and overall computational resources. These limitations affected the extent of hyperparameter tuning, the exploration of more extensive configurations, and the training duration—which was limited to only 5 epochs—potentially impacting the convergence and generalizability of our results. Finally, our study focuses exclusively on adaptor-based methods and LoRA without exploring hybrid approaches or alternative parameter-efficient strategies that might further enhance performance or efficiency.

5.2 Future Work

Building on the current study, future research should address several directions. Hybrid fine-tuning strategies that integrate adaptor-based techniques with LoRA may combine the strengths of both methods to yield an optimal balance between accuracy and efficiency. Extending the experimental analysis to larger, more diverse datasets such as ImageNet or COCO would provide a more rigorous assessment of scalability and generalizability. Moreover, investigating advanced optimization techniques—such as mixed precision training, distributed training, or alternative optimizers—could further reduce training time and memory consumption, particularly for high-rank LoRA configurations. Finally, evaluating these fine-tuning strategies in practical applications, such as object detection, video analysis, or medical imaging, will help determine the feasibility of deploying parameter-efficient Vision Transformers in resource-constrained environments.

6 Conclusion

This study presents a comprehensive comparative analysis of parameter-efficient fine-tuning strategies for Vision Transformers on the CIFAR-10 dataset. Our experimental results, derived directly from complete log outputs, show that while full fine-tuning yields competitive performance (94.93% accuracy, 0.1773 loss), both adaptor-based methods and LoRA dramatically reduce the number of trainable parameters. Notably, adaptor-based approaches (with Bottleneck dimensions from 32 to 128) reduce the training time by approximately 80 s and memory usage by around 0.15 MB compared to the baseline. However, they incur a drop in accuracy (down to 91.54%–91.77%). In contrast, LoRA consistently outperforms the baseline, with the highest configuration (Rank=16) achieving 96.45% accuracy and a lower loss of 0.1095, while maintaining competitive memory usage and inference times.

Based on these findings, we recommend the adoption of LoRA-based fine-tuning in scenarios where accuracy is paramount and modest increases in memory usage are acceptable. For environments with stringent resource constraints, adaptor-based methods offer a viable alternative despite their lower accuracy. Future work should explore hybrid approaches that leverage the strengths of both fine-tuning techniques.

References

[1] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, and Y. Qiao, “Vision transformer adapter for dense predictions,” *ICLR*, 2023.

[2] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, “Parameter-efficient transfer learning for nlp,” *ICML*, 2019.

[3] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” in *ICLR*, 2021.

[4] S. Jie, Z.-H. Deng, S. Chen, and Z. Jin, “Convolutional bypasses are better vision transformer adapters,” *ECAI*, 2024.

[5] N. Ding, X. Lv, Q. Wang, Y. Chen, B. Zhou, Z. Liu, and M. Sun, “Sparse low-rank adaptation of pre-trained language models,” *arXiv preprint arXiv:2311.11696*, 2023.

[6] e. a. Wang, “Factor-tuning: Fine-tuning with low-rank tensors,” *AAAI*, 2022.

[7] G. Luo, M. Huang, Y. Zhou, X. Sun, G. Jiang, Z. Wang, and R. Ji, “Towards efficient visual adaption via structural re-parameterization,” *arXiv preprint arXiv:2302.08106v2*, 2023.

[8] S. Jie, Z.-H. Deng, S. Chen, and Z. Jin, “Convolutional bypasses are better vision transformer adapters,” *FAIA-392-FAIA240489*, 2024.

[9] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, and Y. Qiao, “Vision transformer adapter for dense predictions,” *arXiv preprint arXiv:2205.08534v4*, 2022.

[10] A. Names, “Fact: Factor-tuning for lightweight adaptation on vision transformer,” *arXiv preprint arXiv:25187-Article Text-29250-1-2-20230626*, 2023.

[11] H. e. a. Touvron, “Training data-efficient image transformers & distillation through attention,” in *ICML*, 2021.

[12] W. e. a. Wang, “Pyramid vision transformer: A versatile backbone for dense prediction without convolutions,” in *ICCV*, 2021.

[13] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, 2017.

[14] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 328–339.

[15] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” *arXiv preprint arXiv:1609.04836*, 2017.