Fine-Tuning for Text Classification using Low-Rank Adaptation (LoRA)

Ziming Xu
zix035@ucsd.edu

1 Introduction

LoRA (Low-Rank Adaptation) is a technique designed for parameter-efficient fine-tuning of large pre-trained models. LoRA is a lightweight mechanism to pre-trained models and this method can train model with fewer trainable parameters and reduce computational requirements.

In this project, I try to compare the performance and efficiency of LoRA-based fine-tuning with traditional full-parameter fine-tuning for text classification tasks and explore how the LoRA can implement the efficient fine-tune strategy. Because I don't have sufficient computing resources so I choose this topic. I try to use different dataset and use LoRA-based fine-tuning and traditional full-parameter fine-tuning to explore the difference between their time of training and parameters in training. I also use different parameters in LoRA in order to find how these parameters affect LoRA model.

Based on this exploration, I try some new works to improve the result of datasets. LoRA+ (Hayou et al., 2024) sets different learning rates for adapter matrices A and B, leading to improved performance and faster fine-tuning speeds. And I try to use LoRA+ method in my work to explore weather it can improve it.

2 Related work

Nowadays, many generative AI tools like ChatGPT, Copilot have been used in many situations and make humans life better. These tools are always based on the Large Language Model with a large amount of parameters as training and this way can help model obtain the behavior like human. However, as training these model, because of their parameters, it not only need plenty of time but also hardware resources. It is hard for someone whose resources are limited to train the Large Language Model.

To solve this problem, many works have been proposed like Adapter(Pfeiffer et al., 2020) and Prefix Tuning(Lester et al., 2021). Adapters are layers inserted into the layers of a Transformer model, such as after the attention or feed-forward layers. As fine-tuning, model's other parameters are frozen and just train the Adapters. The number of parameters introduced by adapters is significantly smaller than full fine-tuning so they implement parameter efficiency. Instead of fine-tuning all model parameters, Prefix Tuning use a "prefix" which is a sequence of learnable embeddings and add it to the model's input. As training, base model remains frozen and just train the prefix embeddings.

In my project, I want to explore another fine-tune method, Low-Rank Adaptation (LoRA)(Hu et al., 2021).The author of this work assume that pretrained models have lower intrinsic dimensions when fine-tuned on a specific task. When the model is adaptively fine-tuned, the updated parameter matrix also has a lower intrinsic dimension. A pre-trained model has the parameter matrix $W_0 \in R^{d\times k}$, $\Delta W \in R^{d\times k}$ represents the parameter matrix that changes after fine-tuning. Since the intrinsic dimensionality of $\Delta W \in R^{d\times k}$ is lower(low rank), it can also be expressed as BA, $B \in R^{d\times r}$ and $A \in R^{r\times k}$ and $r \ll min(d, k)$, r can be regarded as the size of rank. Therefore, as fine-tuning, there is no need to fine-tune all parameters, it is enough to just update $\Delta W \in R^{d\times k}$. Now,

$$h = W_0x + \Delta Wx = W_0x + BAx$$

LoRA builds on this idea by replacing these modules with low-rank matrices, achieving even greater parameter efficiency than Adapters.Unlike Prefix Tuning, LoRA don't rely on addressing prompt and can work across a wide range of tasks.

So now LoRA is used in many works and projects, it not only just used in large model languages but also like Diffusion Model(Luo et al., 2024) and so on. There are also some improvement based on LoRA. (Ding et al., 2023) introduces a gating mechanism that allows dynamic adjustments to the intrinsic rank during fine-tuning. These works provide new ways to implement parameter-efficient methods.

LoRA+ (Hayou et al., 2024) introduces a new algorithm to set different learning rates for the low-rank matrices A and B of the model. And this work shows that this method can improve the model relative to the standard LoRA method. It also shows that setting a larger learning rate for matrix B relative to A can enhance feature learning efficiency.

3 Dataset

My project is aim to compare the performance and efficiency of LoRA-based fine-tuning with traditional full-parameter fine-tuning for text classification tasks. So about the dataset, I use the imdb dataset and AG's News dataset.

IMDB Dataset: The Internet Movie Database (IMDB) dataset is a widely used benchmark dataset for binary sentiment classification. It includes movie reviews with positive or negative sentiment labels. There are 25000 samples in training set and 25000 samples in test set. This dataset can be loaded using Hugging Face Library.

I tokenize the text in the dataset and then pad and truncate the tokenized sequences to ensure they have a consistent length and this length is 128.

AG's News Dataset: The AG's News dataset is a benchmark dataset for multi-class text classification. This dataset contains news articles categorized into 4 topics.The 4 topics are World, Sports, Business and Sci/Tech. There are 120,000 samples (30,000 per class) in training set and 7,600 samples (1,900 per class) in test set.

4 Baselines

I use DistilBERT in this project and import "distilbert-base-uncased" pre-trained model through Hugging Face's Transformers library. This model is a distilled version of the BERT.

To implement full-parameter fine-tuning, I set learning rate is 5e-5, train_batch_size and eval_batch_size are both 16, the number of training epoch is 3 and weight decay in regularization is 0.01.

To implement LoRA-based fine-tuning, I use the same setting above and set low-rank dimension is 8, the dropout rate applied to the LoRA layers is 0.1 and lora_alpha is 32.

5 Method

5.1 Experiment Configuration

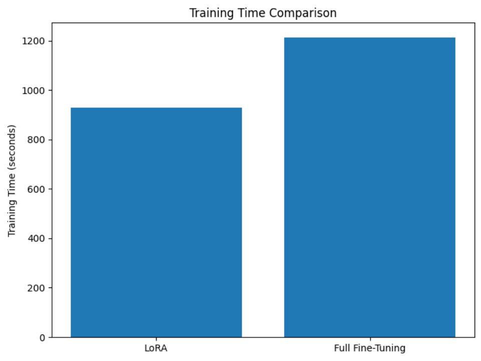
In this project, I use my laptop to finish it with just one NVIDIA GeForce RTX 2060. The project uses Python 3.8.20 as the programming language, with PyTorch 2.4.1 as the deep learning framework, ensuring compatibility with GPU support via CUDA 12.1. The Transformer(4.46.3) libraries from Hugging Face are employed for loading pre-trained models and handling datasets, along with PEFT (0.13.2) for implementing parameter-efficient fine-tuning methods like LoRA.

5.2 Experiment

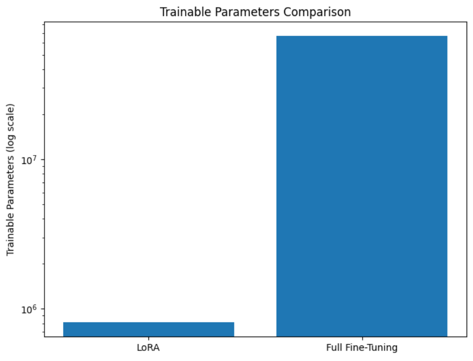
Firstly, I implement the baseline in IMDB Dataset and get the result of training time shown in Figure 1a, training parameters shown in Figure 1b, evaluation loss shown in Figure 1c and accuracy shown in Figure 1d of these two experiments.

It is easily found that using full-parameter fine-tuning and LoRA-based fine-tuning have almost the same evaluation accuracy. However, LoRA-based fine-tuning takes much less training time and training parameters. It is shown that the LoRA-based fine-tuning is a more efficient fine-tune method.

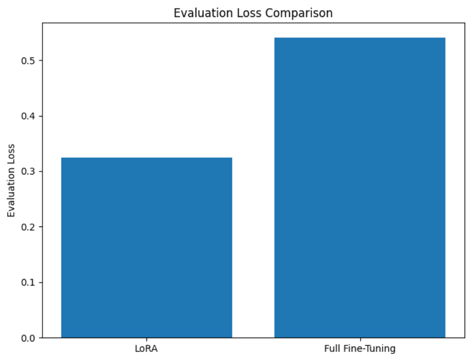
Then, I try to use AG's News Dataset in this experiment. AG's News Dataset is a more complicated dataset. It has 4 categories and the samples are more than which in IMDB Dataset. And the training time shown in Figure 2a, training parameters shown in Figure 2b, evaluation loss shown in Figure 2c and accuracy shown in Figure 2d to compare these two fine-tune methods. Because AG's News Dataset is quiet complicated, it need more time to fine-tune. The visualization shows that using LoRA can also obtain classification accuracy which is close to which obtained by full-parameters fine-tuning. There are lots of dataset with plenty of samples and more complicated categories and it is much more efficient with using LoRA in the large datasets or large models.



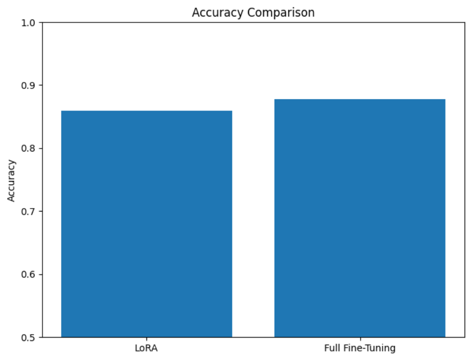
(a) Time Comparison



(b) Parameters Comparison

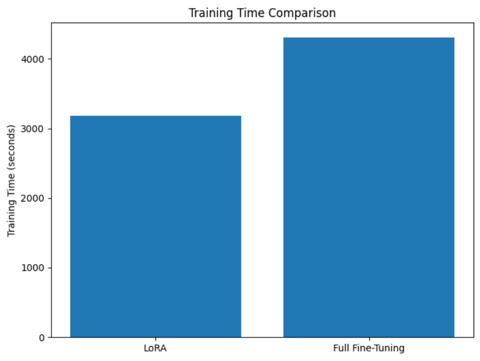


(c) Loss Comparison

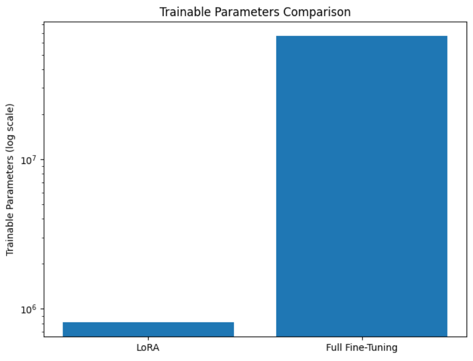


(d) Accuracy Comparison

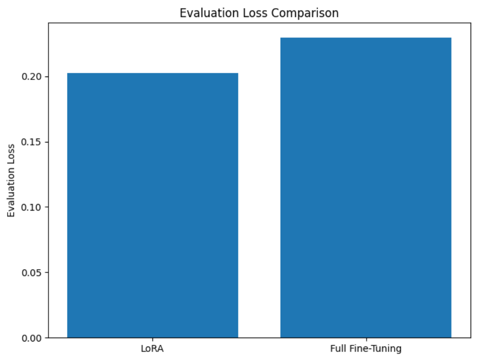
Figure 1: Comparison of IMDB Dataset



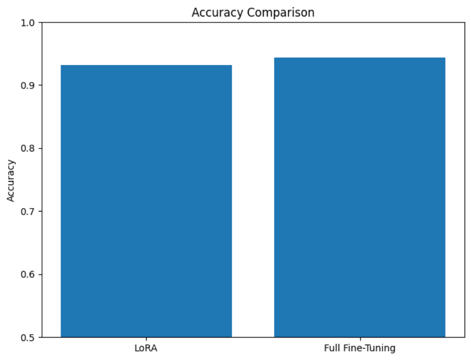
(a) Time Comparison



(b) Parameters Comparison



(c) Loss Comparison



(d) Accuracy Comparison

Figure 2: Comparison of AG's News Dataset

After that, I try to use different hyperparameters of LoRA and obtain the different accuracy and training time based on these hyperparameters. Because these experiments need to repeat many times so I just choose IMDB Dataset to study. Figure 3 shows accuracy, figure 4 shows time consuming and figure 5 shows time and accuracy.

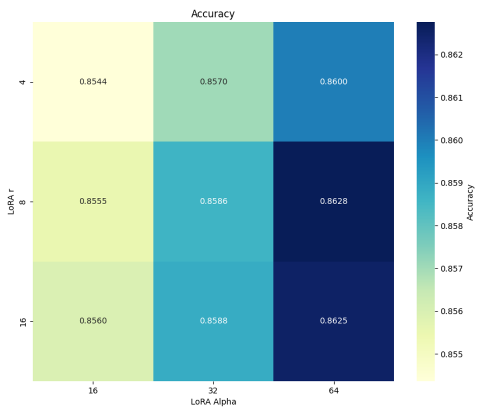


Figure 3: Parameters Comparison

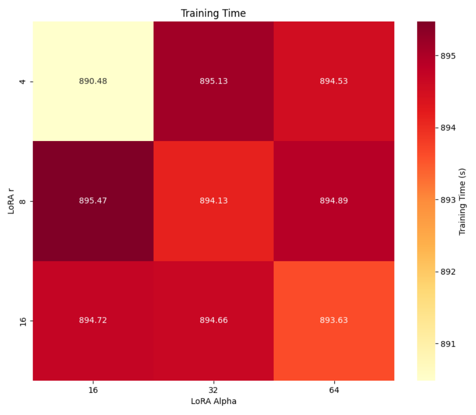


Figure 4: Accuracy Comparison

Finally, I try to use LoRA+(Hayou et al., 2024) in my project with IMDB Dataset. I set learning rate of matrix A is 5e-5 and loraplus_lr_ratio is 2.0 which means learning rate of matrix B is twice that of A.Then I compare its results with normal LoRA in Figure 6. Because in LoRA+, it just change the learning rate of matrix A and B so the total parameters is the same. In the Figure, it shows that LoRA+ is faster than LoRA as getting the similar accuracy.

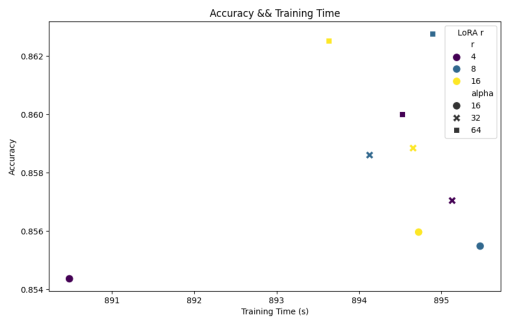
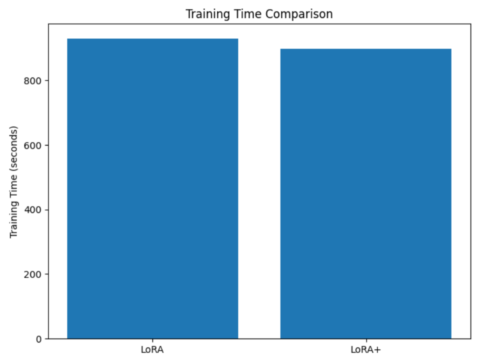
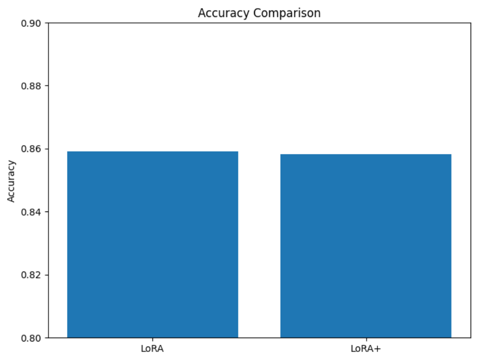


Figure 5: Accuracy Comparison



(a) Time Comparison



(b) Parameters Comparison

Figure 6: Comparison of LoRA and LoRA+

6 Error analysis

Actually, when I use LoRA+ in the project, it doesn't have obvious improvement. The improvement shows in Figure 6 can be implemented by changing the hyperparameters. Because I just use the simple model and simple datasets. If I can have sufficient resource to implement it in the large model and large datasets, the improvement may be more obvious.

7 Conclusion

In the project, I explore the LoRA fine-tuning method and compare it with full fine-tuning method. I visualize the result of these two methods and directly compare the difference of their results. Then I implement the new work of LoRA called LoRA+. In my experiment, I know that using the LoRA fine-tuning method has less parameters than full fine-tuning method, but the difference between their training time is not so big, maybe because the dataset is not very large. Also, I can try to use different learning rate of matrix A and B in LoRA+ and try to confirm that a larger learning rate for matrix B relative to A can enhance feature learning efficiency in larger datasets.

8 Acknowledgements

In this project, I use ChatGPT 40 to help me finish it. I use it to correct my grammar errors in the article. It also provides me some datasets of classificaion to choose with its network-connecting mode. It also helps me to understand the contents of LoRA+ paper. When I try to use LoRA+ in my work, I meet some coding problems and it helps me to solve them.

References

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

Hayou, S., Ghosh, N., and Yu, B. (2024). Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Lester, B., Al-Rfou, R., and 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*, pages 3045–3059.

Luo, Z., Xu, X., Liu, F., Koh, Y. S., Wang, D., and Zhang, J. (2024). Privacy-preserving low-rank adaptation for latent diffusion models. *arXiv preprint arXiv:2402.11989*.

Pfeiffer, J., Kamath, A., Rücklé, A., Gurevych, I., and Cho, K. (2020). Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54.