Deep Learning in Practice 3rd Coding Assignment

Zhongke Sun
zhongke.sun@student-cs.fr

March 5, 2025

1 Experiments Results

1.1 Basic Experiment Setup

- Model: DistilBERT
- Optimizer: AdamW with learning rate $2e-5$ and decay rate 0.01
- Batch Size: 16
- Training Epoch: 1

According to 'Part 6.3: Fine-tuning using the LoRA adapters', 'Part 6.4: Explore the Impact of rank and layers Hyperparameter', 'Part 5: Full-finetuning' and 'Part 6: Fine-Tuning Only the Classification Head', we will use a series of experiments to explore the impact of LoRA adapter and different configurations. As the following table, the first block compares 'Full Fine-Tuning (Baseline)' and 'Fine-Tuning Only the Classification Head'. The second block shows the influence of different ranks when fixing the layer as [query, value, mlp]. Finally, the third block demonstrates the impact of Layers when fixing the rank as 8.

Experiment	Trainable Parameters	Final Loss	Final Accuracy
Full Fine-Tuning (Baseline)	66,955,010	**0.2065**	**91.28%**
Fine-Tuning Only the Classification Head	592,130	0.3988	82.84%
Fine-Tuning using the LoRA adapters	1,108,226	0.2661	88.22%
Impact of Rank (Fixed Layers = [query, value, mlp])			
Rank = 2	721,154	0.2951	86.97%
Rank = 8	1,108,226	**0.2585**	**89.50%**
Rank = 32	2,656,514	0.2970	86.78%
Impact of Layers (Fixed Rank = 8)			
Layers = [query, value]	739,586	**0.2450**	**89.62%**
Layers = [query, value, mlp]	1,108,226	0.2595	89.41%
Layers = [query, value, key, proj, mlp]	1,255,682	0.2581	89.28%

Table 1: Comparison of Performance and Trainable Parameters for Different LoRA Configurations

2 Observations

2.1 How do the number of parameters and performance compare to full fine-tuning?

Full fine-tuning achieves the best performance but requires training the entire model (around 67 million parameters), making it computationally expensive.

Only fine-tuning the classification head drastically reduces the number of trainable parameters (Only 592K) but leads to a significant drop in accuracy (-8.44%).

LoRA significantly reduces trainable parameters compared to full fine-tuning (1.1M vs. 67M) but still maintain a good accuracy (88.22%).

2.1.1 LoRA with Fixed Layers = [query, value, mlp] and Different Ranks

Rank = 8 provides the best trade-off, reaching **89.50%** accuracy and only 1.1M parameters (98.4% fewer than full fine-tuning). Increasing rank to 32 does not improve performance but increases parameter count.

Rank = 8 is the optimal choice for ranking setup.

2.1.2 LoRA with Fixed Rank = 8 and Different Layers

Using **only query and value layers** achieves the best performance (**89.62% accuracy, 0.2450 loss**) with just **739K** parameters. Adding MLP increases parameters to 1.1M but does not significantly improve accuracy. Adding more layers further increases parameters (1.25M) but no benefit for model performance.

LoRA applied only to query and value layers is the most parameter-efficient setup.

2.2 What is the trade-off between parameter efficiency and performance?

There are few things about the trade-off between parameter efficiency and performance.

2.2.1 Full Fine-Tuning vs. LoRA

Full fine-tuning achieves the highest accuracy (91.28%) and lowest loss (0.2065) but requires train all 67M parameters. LoRA (Rank=8, Layers=[query, value]) reduces parameters by 98.9% while maintaining 98.2% of full fine-tuning accuracy (89.62%).

This makes LoRA a much more efficient alternative with minimal accuracy drop.

2.2.2 Higher Ranks vs. Lower Ranks - Increasing Rank Beyond a Specific Number Does Not Improve Performance

When fixing the layers, Rank=8 is the optimal trade-off (1.1M parameters, 89.50% accuracy). Rank = 32 significantly increases parameters (2.6M) but actually results in worse accuracy (86.78%), suggesting diminishing returns.

So the key trade-off is to find the optimal rank. Too bigger rank or too smaller rank will do harm to the model performance.

2.2.3 More Layers vs. Less Layers

Using query and value layers alone (739K parameters) outperforms adding mlp or key/proj layers.

Adding more layers increases the trainable parameters but does not improve accuracy, which means that the computational cost increases without meaningful benefits.

2.2.4 Extreme Low cost vs. Slightly Allow More Cost - Fine-Tuning Only the Classification Head Has the Worst Trade-off

Training only the classification head (592K params) leads to a large accuracy drop (82.84%), proving that adapting deeper transformer layers (even partially, like in LoRA) is necessary for good generalization. We just need to allow a bit more parameters to get a dramatic improvement. (82.84% $\rightarrow$ 89.62%)

The best trade-off is LoRA (Rank = 8, Layers = [query, value]). 98.89% fewer parameters than full fine-tuning with only 1.81% accuracy loss.