Parameter-Efficient Fine-Tuning of DistilBERT for Sentiment Classification on IMDB

Authors: Yoong Jun Han, gpt-5(for LaTeX code generation/clean up, summary)

***Abstract*—I investigate the adaptation of large pretrained Transformer models to a domain-specific sentiment classification task. I compare full fine-tuning (FT) of DistilBERT against parameter-efficient fine-tuning using LoRA (Low-Rank Adaptation). Experiments are conducted on the IMDB dataset with 25k training and 25k evaluation samples, balanced across positive and negative classes. I analyze performance, training efficiency, and loss dynamics across different dataset sizes, providing insights into trade-offs between adaptation strategies. All code required to reproduce this report is available at https://github.com/yjh-jy/dsa4213-assignment3-finetuning-transformers.**

I. Introduction

Large pretrained Transformers such as BERT and its variants have demonstrated strong performance across NLP tasks. However, full fine-tuning can be computationally expensive, particularly for resource-constrained environments. Parameter-efficient tuning methods, such as LoRA, aim to reduce the number of trainable parameters while retaining most of the performance (1). In this work, I investigate these strategies for sentiment classification on IMDB movie reviews, exploring trade-offs in accuracy, loss dynamics, and compute efficiency. I leverage DistilBERT due to its smaller size and suitability for limited hardware such as an Apple M1 Pro chip.

II. Dataset

I use the IMDB sentiment dataset available via HuggingFace (2). The dataset consists of 50k movie reviews, balanced evenly between positive and negative labels. I utilize 25k samples for training and 25k for evaluation. Each review is tokenized and truncated to a maximum sequence length of 256 tokens.

TABLE I
IMDB Dataset Statistics

Split	Samples	Class Balance (%)
Training	25,000	50/50
Evaluation	25,000	50/50

III. Methods

A. Model Choice

I select **DistilBERT** for its computational efficiency and competitive performance on classification tasks. Its smaller parameter count (67.7M) allows full fine-tuning and LoRA adaptation on limited resources without prohibitive memory requirements. Following the approach in (1), LoRA is applied to attention query and value matrices, effectively capturing task-specific information while freezing the majority of pre-trained parameters.

B. Training Setup

1) Common Setup: All experiments use the **HuggingFace Trainer API** for consistent training and evaluation, allowing automatic device placement, batch processing, checkpointing, and logging. Metrics tracked include macro F1, accuracy, training and evaluation loss, and per-step training time. All experiments are conducted on a single GPU (or Apple M1 Pro) to reflect realistic resource constraints.

2) Full Fine-Tuning (FT):

- **Trainable Parameters:** All 67.7M parameters of DistilBERT are updated.
- **Batch Size:** 32, same as LoRA to ensure fair comparison of efficiency and performance.
- **Learning Rate:** 5e-5, standard for DistilBERT classification tasks.
- **Optimizer:** AdamW, robust for transformer architectures with weight decay.
- **Epochs:** 3, constrained by computational budget and time while providing sufficient exposure to training data.

Justification: Full FT allows the model to fully adapt to the sentiment classification task, capturing fine-grained patterns in the training data. The conservative learning rate prevents divergence given the large number of trainable parameters.

3) LoRA Fine-Tuning:

- **Trainable Parameters:** Approximately 739K (1.1% of full model) including low-rank adaptation matrices in attention layers and classifier layers.
- **Batch Size:** 32. This is in line with (3), where their research suggests that LoRA is less tolerant of large batch sizes (128, 256) compared to FullFT. I also did not have the GPU memory for higher batch sizes so this was a natural choice.
- **Learning Rate:** 1e-4, higher than FT to compensate for smaller trainable parameter count and enable faster adaptation. This is in line with (3), which notes that the optimal LR for LoRA is at least 10x higher than Full FT due to the 1/r scaling factor.
- **Optimizer:** AdamW, consistent with FT for controlled comparison.
- **LoRA Configuration:** Rank $r = 8$ applied to attention query and value projection layers across all 6 transformer layers.
- **Epochs:** 3, same as FT for comparability.

Justification: LoRA constrains parameter updates to a small subset, resulting in more stable training and lower evaluation loss. A higher learning rate ensures meaningful adaptation within a limited parameter space, while batch size and epochs are kept consistent with FT.

IV. Results

A. Loss Curves

Figure 1 shows training and evaluation loss curves for FT and LoRA across all dataset sizes. The training steps are normalized to account for variations between the dataset sizes. Several observations can be made:

- For **full fine-tuning (FT)**, the evaluation loss consistently remains above the training loss throughout training, suggesting potential overfitting to the training data.
- For **LoRA**, training and evaluation losses are well-aligned at all dataset sizes, reflecting a regularization effect from freezing most pretrained parameters.
- LoRA's evaluation loss is lower than FT's at corresponding dataset sizes, even though its macro F1 performance is slightly lower. This suggests that LoRA minimizes loss effectively but lacks the full expressive power of FT for discriminative performance.
- The gap between training and evaluation loss decreases as dataset size increases, indicating that larger datasets partially mitigate overfitting.
- LoRA exhibits more stable loss curves with smaller fluctuations per step (other than the initial one), likely due to its smaller number of trainable parameters while FT experience more and larger fluctuations. LoRA shows a sharper curves near the start which eventually tapers off but FT continues the downward trend more aggressively, breaking off from its evaluation curve near the midway mark.

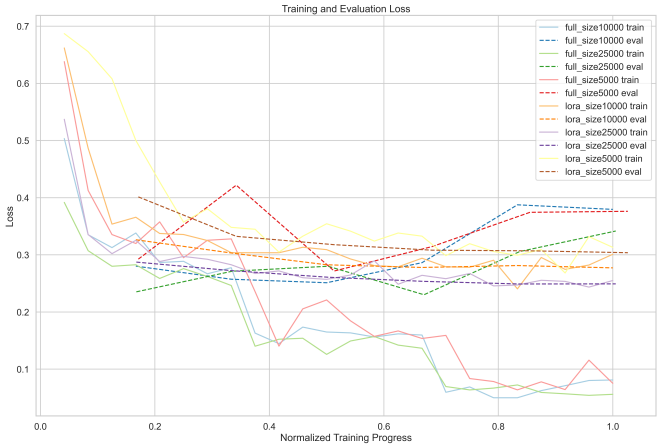

Fig. 1. Training and evaluation loss curves for FT and LoRA across 3 dataset sizes: 25k, 10k, and 5k.

B. Performance Metrics

Table II and Figure 2 summarize macro F1 scores across dataset sizes for both FT and LoRA.

Observations:

- **Full Fine-Tuning (FT)** consistently outperforms LoRA across all dataset sizes.
- **LoRA** achieves competitive results using only ∼1% of the parameters, demonstrating its parameter efficiency.
- Increasing dataset size improves performance for both strategies, but the relative gap remains around 2–3% macro F1.
- These results align with the loss trends: LoRA provides stable generalization, whereas FT benefits more from larger datasets.

TABLE II
Macro F1 Scores for FT and LoRA

Strategy	Dataset Size	Macro F1
FT	5,000	0.895
FT	10,000	0.905
FT	25,000	0.916
LoRA	5,000	0.869
LoRA	10,000	0.882
LoRA	25,000	0.897

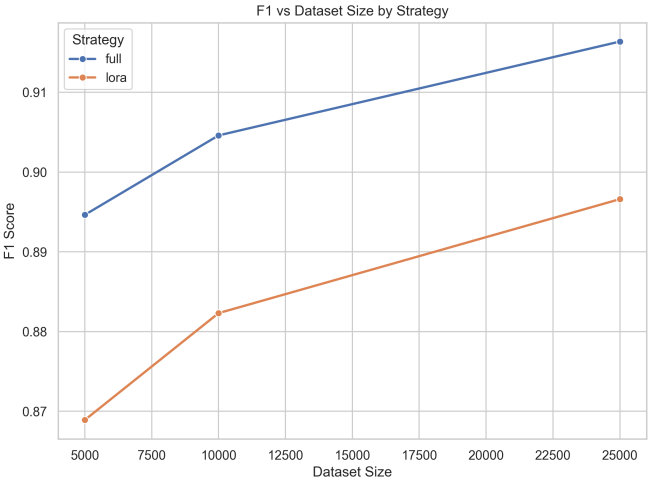

Fig. 2. Macro F1 versus dataset sizes for FT and LoRA.

C. Time Taken

Table III illustrates training time (including evaluation overhead) for FT and LoRA.

Observations:

- LoRA updates far fewer parameters, yet total training time is comparable to FT due to evaluation overheads at smaller datasets sizes. This is because I evaluated twice an epoch and each evaluation takes a full pass at the 25K test dataset, taking around 3-4 mins, potentially dominating the training time in smaller datasets. For larger datasets, LoRA and FT starts to differentiate themselves with LoRa being 3% faster than FT at 10K and 9% faster than FT at 25k.
- Overall, LoRA achieves similar or better wall-clock efficiency while greatly reducing trainable parameters. A

note to be taken is that the Total FLOS (Total FLOPS) measured by the Trainer API paints a different story, with the numbers for LoRA being roughly similar or even exceeding that of FT despite training fewer parameters. This is taken to be an anomaly that I will not discuss in this report.

TABLE III
TRAINING TIME TAKEN FOR FT AND LORA

Strategy	Dataset Size	Time (min)
FT	5,000	27.8
FT	10,000	35.2
FT	25,000	56.8
LoRA	5,000	28.2
LoRA	10,000	34.2
LoRA	25,000	51.8

D. Parameter Efficiency

To better understand how efficiently each method utilizes trainable parameters, I computed *F1 per million trainable parameters*, defined as:

$$\text{F1 per M params} = \frac{\text{Macro F1}}{\text{Trainable Parameters}/10^6}$$

Table IV and Figure 3 illustrate this metric for both full fine-tuning (FT) and LoRA across dataset sizes.

TABLE IV
F1 PER MILLION TRAINABLE PARAMETERS

Strategy	Dataset Size	Trainable Params	F1 per M Params
FT	5,000	67.7M	0.0132
FT	10,000	67.7M	0.0134
FT	25,000	67.7M	0.0135
LoRA	5,000	0.74M	1.175
LoRA	10,000	0.74M	1.1930
LoRA	25,000	0.74M	1.2123

Observations: LoRA exhibits over two orders of magnitude improvement in parameter efficiency compared to full fine-tuning. Despite slightly lower absolute F1, LoRA delivers nearly 90× higher F1 per million parameters. This efficiency remains consistent across dataset sizes, making LoRA highly practical in resource-constrained or multi-task scenarios.

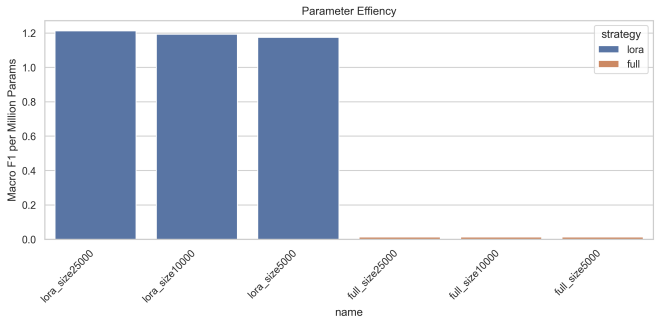

Fig. 3. F1 per million trainable parameters for FT and LoRA across dataset sizes. LoRA achieves ∼90× higher efficiency.

V. LIMITATIONS

There are a few limitations with this report I would like to point out and clarify:

- **Lack of long term training:** I only trained for 3 epochs and this is could be insufficient for LoRA and FT to demonstrate their respective advantages and disadvantages over a longer time horizon.
- **Lack in variations of learning rate for LoRA:** In (3), they demonstrated that the optimal LR should be set higher for shorter training runs and varied as training progressed. Preliminary evidence suggests an optimal multiplier around 15x over the FullFT for short runs. I only used a fixed, 10x higher LR than FT for the LoRA runs so this could be preventing LoRA in my runs from reaching its full potential.
- **Lack in variations in the layers LoRA is applied:** As mentioned in (3), they demonstrated that when LoRA is applied to all layers especially the MLP/MoE layers, LoRA shows a similar training dynamics to FT at the very start of training. I did not vary the layers in which LoRA was applied and only stuck to the attention layers so that could be another reason why LoRA was still lagging behind FT in terms of Macro F1 by 2-3%.

VI. CONCLUSION

I demonstrate that DistilBERT can be effectively adapted for sentiment classification using both full fine-tuning and LoRA. While FT consistently achieves higher absolute accuracy, LoRA offers a remarkably parameter-efficient alternative with stable training dynamics and lower evaluation loss. My analysis highlights trade-offs between parameter efficiency, compute cost, and task performance, providing practical insights for fine-tuning under constrained computational budgets.

REFERENCES

[1] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, F. Wang, L. Wang, W. Chen, and B. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," *ICLR 2022*, https://arxiv.org/abs/2106.09685

[2] "IMDB Dataset of 50K Movie Reviews," *HuggingFace*, https://huggingface.co/datasets/stanfordnlp/imdb

[3] Schulman, John and Thinking Machines Lab, "LoRA Without Regret," *Thinking Machines Lab: Connectionism, Sep 2025.*, https://thinkingmachines.ai/blog/lora/