Fine Tuning with LoRA
Text Classification

Alex Gonzalez, Chloe Kim, Richard Zhong

New York University, New York, USA
https://github.com/Adgonzalez2018/DL-Project2

Abstract

This project aims to develop a fine-tuned text classification model to categorize news articles into categories: World, Sports, Business, or Sci/Tech, while adhering to the strict constraint of no more than 1 million trainable parameters. Using **Low-Rank Adaptation** (LoRA) on a pre-trained **RobERTa** model, we explored the impact of different hyperparameter configurations– including matrix rank, scaling strength, and dropout to optimize model performance under the constraint. The adapted model with parameter constraints achieved an accuracy of **85.55%**. These findings contribute to the broader effort of designing cost-effective alternatives to full model fine-tuning, especially for resource-constrained applications of large language models in classification tasks.

Introduction

Parameter-Efficient Fine-Tuning (PEFT) offers a cost-effective solution to the resource-intensive process of fine-tuning large language models (LLMs). The idea is to adjust only a small subset of parameters to adapt the pre-trained model for a task rather than to update all parameters of a pre-trained model. This significantly reduces computational cost and memory usage. A popular and effective method for **PEFT** methods is **Low-Rank Adaptation** (LoRA), which introduces trainable rank-decomposed matrices into existing layers. This approach is particularly well-suited for large transformer-based models applied to tasks like text classification.

In this project, we apply **LoRA** to fine-tune a pre-trained **RoBERTa** model on the **AG News dataset**, aiming to classify news headlines into one of four categories. While **LoRA** allows for efficient adaptation, we explore the limitations of fine-tuning alone, especially when evaluating the model on a structurally different test provided through a Kaggle competition. Notably, this unlabelled dataset contains longer and more complex text samples, which introduces a distribution shift that challenges the generalization capacity of our fine-tuned model.

To address this, we investigate various strategies beyond basic fine-tuning, including inference chunking, dropout tuning, label smoothing, and inference-time data augmentation. Our goal is to understand how parameter-efficient models behave under distributional shifts and to evaluate how different techniques impact the model's robustness and accuracy.

LoRA

Low-Rank Adaptation is an advanced fine-tuning technique that adapts large language models to new tasks by introducing a small set of trainable parameters. Memory usage is significantly reduced since the pre-trained weights are frozen and only the weights with low-rank matrices are trained before merging. This approach with fewer trainable parameters means gradient computation becomes easier so computation cost is also reduced. It is effective for fine-tuning LLMs in text generation, translation and question answering.

LoRA avoids updating the full weight matrix of a pre-trained large language model by fine-tuning two smaller matrices to approximate the original weights. After training, the two low-rank matrices are integrated into the pre-trained model for inference. As shown in Figure 1, the pretrained weights are represented as the sum of the pretrained weights (W_0) and only the low-rank matrices, blocks A and B are trained.

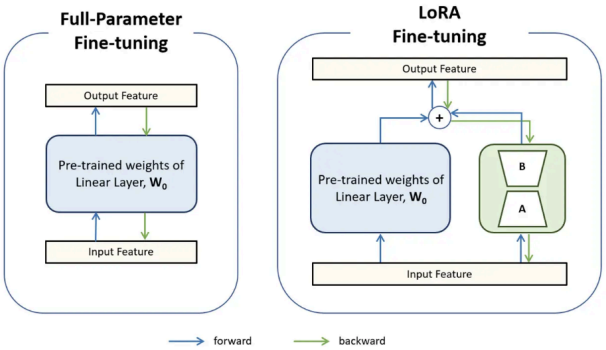

Figure 1: Block diagram of Fine-tuning models

The two hyperparameters that can be experimented with are the rank of the matrix AB, denoted by r, and the strength, denoted by α. As rank r is increased, so does the number of parameters that need to be updated during adaptation. Smaller r can lead to computationally efficient training, but a lower quality of the model. However, increasing r beyond a certain threshold may not yield any discernible improvement in the quality of the model output. The strength α is a scalar multiplied by AB when added to the original weight. When minimal alteration to the base model weights is desired, smaller values α are chosen, and vice versa. These knobs can be adjusted per layer and per weight matrix to gain accuracy.

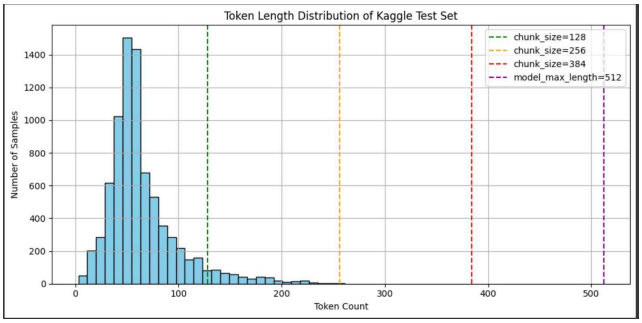

Figure 2: Token Length Distribution of Kaggle Test Set with Chunk Size Thresholds

Methodology

We fine-tuned a pre-trained Roberta-base model using Parameter-Efficient Fine-Tuning (PEFT) with **Low-Rank Adaptation**. **LoRA** was applied to the attention layers by injecting low-rank matrices into the query and value projections. We experimented with different ranks and scaling factors. While staying below the 1 million trainable parameter constraint.

The model was trained on the **AG News dataset**, which consists of a 4-way classification of news headlines into World, Sports, Business, and Sci/Tech. We used a learning rate of 2e-4, a batch size of 16, and trained for up to 5 epochs. Several optimizers were also evaluated during early experimentation, with AdamW proving more stable and consistent across runs.

Experiments & Analysis

LoRA Hyperparameter Sweep (r, α, Dropout, Optimizer)

We experimented with several combinations of **LoRa** rank (r) and scaling factor (α) to identify the most effective configuration. Our tuning focused primarily on values of r ranging from 5 to 9, with α typically set to 2 x r. We found that r = 8 and α = 16 offered the best balance between model capacity and generalization, consistently delivering the strongest performance on the Kaggle dataset.
In addition, we briefly explored multiple optimizers. While Adafactor yielded slightly better accuracy on the **AG News** validation set, AdamW (Adamw_torch) consistently performed better on the Kaggle test set and was therefore used in our final model submissions going forward.

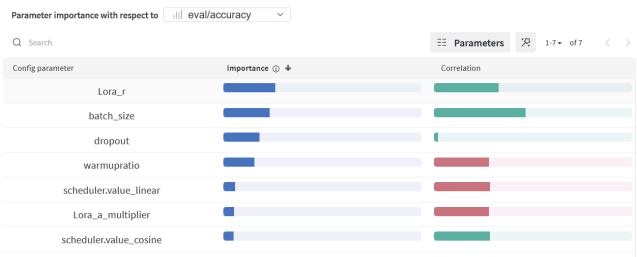

Figure 3: Hyperparameter Importance and Correlation with Validation Accuracy

Figure 3 shows the results of our parameter sweep of 52 runs, with each run training on 1200 examples to save time while providing indications for optimal hyperparameter values. Parameters tested were: **LoRA** r value, **LoRA** alpha multiplier (alpha = 1 x r or 2 x r), training batch size, **LoRA** dropout, cosine vs. linear learning rate scheduling, and learning rate warmup ratio. The importance and strength of each parameter's positive or negative correlation with evaluationrf accuracy are shown. All runs were using the AdamW optimizer.

Model Configuration	Validation Accuracy	Kaggle Accuracy
LoRA (r=6, α=12)	94.06%	84.58%
LoRA (r=7, α=14)	94.68%	83.75%
LoRA (r=8, α=16)	93.44%	**85.55%**
LoRA (r=9, α=18)	93.28%	84.8%
Chunking (128 tokens)	93.48%	84.1%
Label Smoothing (ϵ=0.1)	93.75%	83.58%

Table 1: Validation & Kaggle Accuracies for Different **LoRA** Hyperparameter Settings.

Analyzing Dataset Characteristics

After identifying our best-performing model through a comprehensive hyperparameter sweep, we found that further tuning efforts failed to improve accuracy beyond our initial baseline. This prompted us to shift focus from model optimization to analyzing the data itself.
By comparing the **AG News** training set and the unlabeled Kaggle test set, we observed a significant difference in text length distributions. The **AG News** consisted of a generally shorter title/headline, while the Kaggle dataset contained much longer texts depicted in Figure 2. This mismatch in input length highlighted a potential distribution shift that could explain why our model, despite performing well on validation data, struggled to generalize further on the unseen test set.

Inference Chunking

Inference chunking is a strategy used to handle longer text inputs at inference time by breaking them into smaller overlapping segments (chunks). This is particularly useful when the model was trained on shorter sequences (e.g., **AG News** headlines) but is applied to longer, more complex samples, as seen in the Kaggle test set. Chunking allows the model to process multiple overlapping segments and aggregate predictions (e.g., by averaging logits or using majority voting). This can improve robustness and ensure important information from later parts of the text isn't ignored.

We experimented with different chunk sizes to evaluate how granular segmentation affects classification performance:

- **384 tokens**: Serves as a baseline, since it matches the max_length during training.
- **128 tokens**: Smaller segments to capture finer-grained signals across the input.
- **64 tokens**: Very fine granularity, potentially useful for catching localized cues, but also introduces more noise and redundancy.

Despite the conceptual advantages, we observed that chunking did not significantly improve accuracy over standard inference using the full input. Even with smaller chunk sizes. Depicted in Table 2.

Data Augmentation

We tried three strategies to augment the Kaggle test data in hopes that it would allow the model to correctly classify these longer examples. Note that these augmentations were applied only at inference time and served as a lightweight attempt to mitigate the impact of distribution shift without retraining the model. Each approach aimed to reshape long inputs to better reflect the structure and length of the **AG News** training samples:

1. **Random 50-word sample:** For long examples, we randomly selected 50 words, aligning with the average length of training data
2. **First 50 words:** Assuming key context appears early, we used the first 50 words of each long input.
3. **Random contiguous span:** To preserve semantic flow, we sampled a random block of 50 consecutive words.

Augmentation Strategy	Kaggle Accuracy
Chunking (128 tokens)	84.1%
Label Smoothing (ϵ=0.1)	83.58%
Random 50-word Sample	83.60%
First 50 Words	83.45%
Random Contiguous Span (50 words)	83.78%

Table 2: Kaggle Accuracy for Various Inference-Time Data Augmentation Techniques

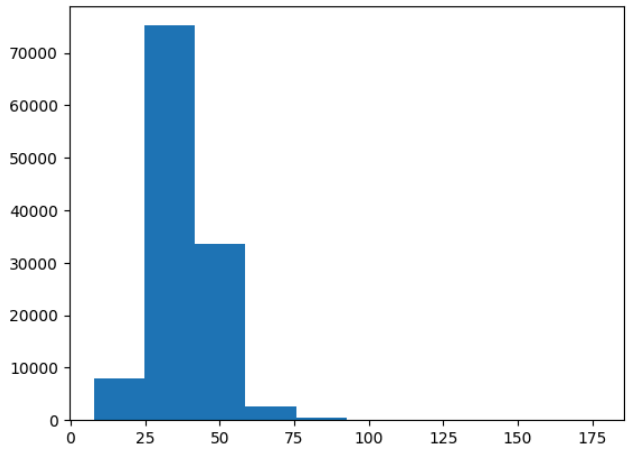

Figure 4: Distribution of training examples with respect to word count

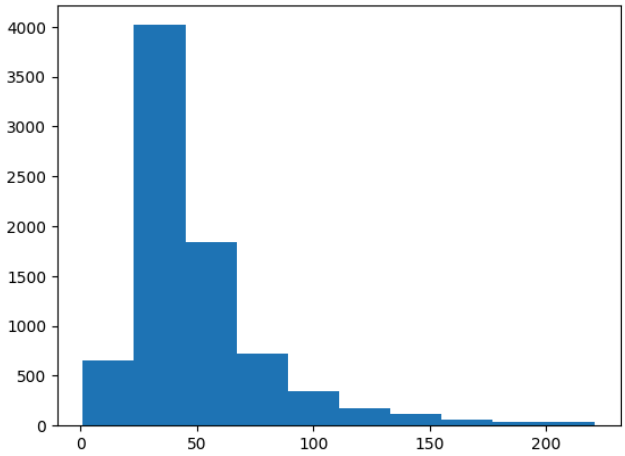

Figure 5: Distribution of Kaggle test examples with respect to word count

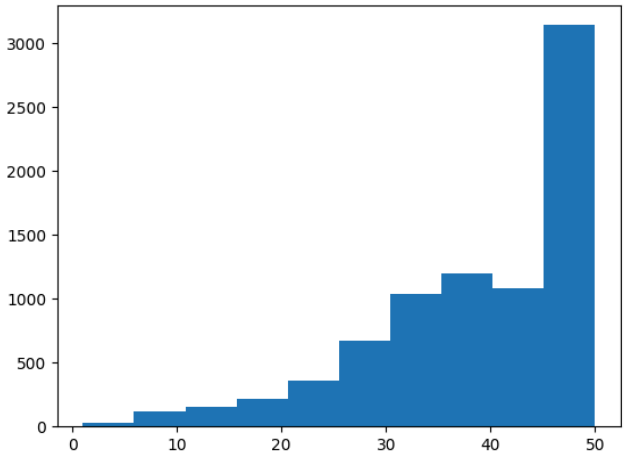

Figure 6: Distribution of test examples after data augmentation

However, none of these strategies improved our final test score, indicating that there may be some model-specific issues that were keeping us from benefiting from these augmentations.

Results/Discussion

Our hyperparameter sweep confirmed that increasing **LoRA's** rank and scaling factor beyond a certain point did not yield consistent results. The configuration of r = 8 and α = 16 consistently offered the strongest generalization on the Kaggle test set, achieving our highest private score of 85.55%.

Inference-time strategies such as chunking, label-smoothing, and lightweight data augmentations were tested to address the distribution shift between our training dataset and the private unlabelled dataset. Despite promising hypotheses, none of these augmentations meaningfully outperformed the base model, with most scoring between a range of 83-84%.

This indicates that the model's robustness is more strangely tied to the training-phase alignment of input structure and semantic coverage than to superficial post-hoc fixes. Furthermore, we observed that improvements on the **AG News** validation set didn't always translate to gains in the private Kaggle test set– underscoring the challenge of generalizing across datasets with subtle but impactful structural differences (e.g., text length, topic coverage).

Conclusion

In this project, we successfully fine-tuned a Roberta-base model using **LoRA**, remaining within a strict constraint of under 1 million trainable parameters. Through an extensive hyperparameter sweep, we identified an optimal configuration that balanced performance and efficiency, achieving a private leaderboard score of 85.55% on the Kaggle dataset.

While we explored various inference-time techniques to close the generalization gap caused by distribution shift, these approaches offered only marginal improvements at best. Our findings suggest that while **LoRA** is highly effective for parameter-efficient adaptation, meaningful gains on structurally different datasets may require training-aware adjustments (e.g., exposure to long-form text) rather than post-hoc augmentation alone.

Acknowledgment

We acknowledge the use of OpenAI's ChatGPT for certain sections of the report.

References

Xu, L., Xie, H., Qin, S., Tao, X., & Wang, F. L. (2023). Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment. *ArXiv*. https://arxiv.org/abs/2312.12148

Hu, E. J., Shen, Y., Wallis, P., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models. *ArXiv*. https://arxiv.org/abs/2106.09685

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*. https://arxiv.org/abs/1907.11692