# Project Report: Finetuning with LoRA

**loaded**
**Aryaman Singh Dev, Nevin Mathews Kuruvilla, Rohan Subramaniam**
**New York University - Tandon School of Engineering**
**{asd9884, nm4709, rs9194}@nyu.edu**
**Github repository: `https://github.com/NevinTroy/Roberta_PEFT`**

## Abstract

This paper aims to present a detailed analysis of our LoRA fine-tuned Roberta model on the AG News dataset for a text classification task. We have designed an optimal PEFT (Parameter Efficient Fine Tuning) configuration to get the fine tuned model. Our model achieves a final evaluation accuracy of 90.04% on the news dataset with 907,012 trainable parameters. This report discusses our training methodology along with recommendations for further improvements. This analysis demonstrates how effectively fine tuning on a small fraction of the parameters can reduce training time and space while not dropping the accuracy.

## Introduction

Text classification is a fundamental task in Natural Language Processing (NLP), enabling the categorization of text into predefined labels. Transformer-based models, particularly RoBERTa (a robustly optimized BERT pretraining approach), have shown state-of-the-art performance on various classification tasks due to their ability to capture contextual relationships effectively.

In this project, we fine tune the Roberta base model using the LoRA technique. LoRa dramatically reduces the number of trainable parameters by freezing the model weights and updating the weights with low rank matrices.

The AG News dataset contains over 120,000 news articles from over 2000 news sources across the web, classified into 5 classes: World, Sports, Business, Sci/Tech.

This report additionally compares different PEFT configurations and how they affect model performance, while making sure the number of trainable parameters are less than 1,000,000.

## Data Pipeline and Preprocessing

Our data pipeline begins with loading the AG news dataset from hugging face.

- **Data Loading:** We load the AG news dataset from hugging face. Explicit label mapping is applied to the classes to ensure consistency between training and inference.
- **Data Splitting:** We divide training data into training (90%) and validation (10%) subsets.

## Model Architecture

Our transformer-based architecture leverages the RoBERTa encoder (M2) for downstream text classification:

### Base Model

We use the pretrained `roberta-base` model from HuggingFace's Transformers library as the backbone. It includes:

- **Embedding Layer:** Word, position, and token type embeddings, each of dimension 768
- **Encoder:** 12 transformer layers, each with:
  - Multi-head self-attention (12 heads)
  - Feed-forward network with hidden size 3072 and GELU activation
  - Layer normalization and dropout after each sub-layer

### Classification Head

We append a lightweight feed-forward classification head to fine-tune on our specific task:

- **Intermediate Layer:** Linear layer projecting from 768 to 768 dimensions, followed by ReLU and dropout
- **Output Layer:** Linear layer mapping to 4 output classes
- **Initialization:** The classification head weights are randomly initialized and trained from scratch

### Regularization and Optimization

- **Dropout:** Applied in both the transformer blocks and classification head with a probability of 0.1
- **Fine-tuning:** All RoBERTa layers are fine-tuned end-to-end using cross-entropy loss

This architecture retains RoBERTa's powerful contextual encoding while adapting the output head to suit multi-class classification tasks (4 classes in our case).

## Training Methodology

For the RoBERTa-based PEFT (Parameter-Efficient Fine-Tuning) model, we adopt a training regime tailored for transformer finetuning with LoRA modules:

### Training Strategy

- **Base Model:** `roberta-base`, integrated with LoRA adapters targeting attention layers (`query` and `value` projections)
- **PEFT Configuration:** Only 0.72% of the total parameters are trainable (907K of 125M), allowing efficient adaptation

### Optimization

- **Optimizer:** AdamW with learning rate of 1e-4 and weight decay of 0.01
- **Gradient Clipping:** Maximum gradient norm set to 1.0
- **Gradient Accumulation:** Accumulation steps set to 4 to simulate a larger effective batch size
- **Warmup:** Linear warmup with 10% of total steps

### Learning Rate Scheduling

- **Scheduler:** Linear decay scheduler
- **Warmup Ratio:** 0.1 (10% of training steps used for linear warmup)

### Training Configuration

- **Epochs:** 6
- **Batch Size:** 32 (training), 64 (evaluation)
- **Precision:** Mixed precision training (FP16) enabled when using a GPU
- **Evaluation Strategy:** Periodic evaluation every 200 steps
- **Checkpointing:** Model checkpoints saved every 200 steps; best model loaded at the end based on validation accuracy
- **Seed:** 37 (for reproducibility)
- **Miscellaneous:** Logging every 50 steps; gradient checkpointing enabled to reduce memory usage

### Loss Function and Metrics

- **Loss:** CrossEntropyLoss
- **Evaluation Metric:** Accuracy used to select the best model

This setup ensures efficient fine-tuning of large-scale language models with limited compute by leveraging LoRA adapters, while maintaining competitive performance through careful optimization and checkpointing.

## Results and Analysis

### Model Performance

Our model achieved a local evaluation accuracy of 90.04% accuracy on the test set and a public score accuracy of 84.65% on an unseen test dataset while maintaining a parameter count of 907,012 , well within the 1M constraint.

Table 1: Model Performance Summary

| Metric | Value |
|---|---|
| Parameters | 907,012 |
| Local Test Accuracy | 90.04% |
| Public Test Accuracy | 84.65% |

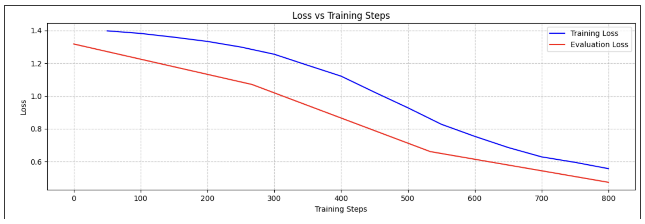

Figure 1: Training Loss and Evaluation Loss plotted over training steps

### Advantages of Final approach

Parameter efficiency by using only 0.72% of total parameters. Training efficiency by reducing memory requirements by reducing parameters. Unlike adaptor based approaches, the final model doesn't introduce inference latency as low-rank matrices can be merged with frozen weights at inference time.

## Key learning

- **Adapter Coverage:** Targeted adaptation of specific attention components (query and value only) resulted with less number of parameters; provided the best performance-to-parameter ratio. Including key matrices increased number of parameters without performance gains. Adding `output.dense` and `intermediate.dense` layers improved performance but at a higher parameter cost.
- **Optimizer choice:** AdamW with low learning rate (1e-4) and marginal weight decay (0.01) outperformed the SGD optimizer, that showed instability. Linear learning rate schedules with warmup improved convergence.
- **Batch sizes and Training steps:** Smaller batch sizes for training allowed more update steps. Larger batch sizes evaluated improved efficiency.
- **Adaptation vs Generalization:** Models with fewer training parameters sometimes showed better generalization to unseen data, suggesting overfitting due to parameter-efficient methods.
- **LoRA rank selection:** The expressivity of the adaptation determined by the rank. Lower work well for simple classification tasks, while higher provided good performance on more complex tasks. Ranks beyond 10 showed diminishing returns.

## Recommendations for Further Improvement

To further enhance the performance and efficiency of the RoBERTa-based PEFT model, we propose the following avenues for improvement:

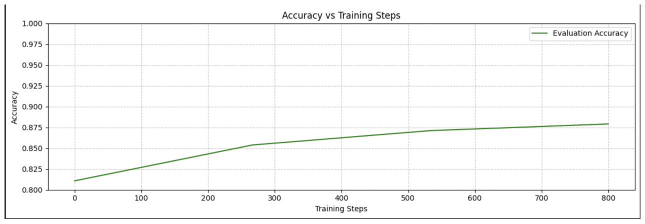

Figure 2: Accuracy over Training steps

- **Increased Adapter Coverage:** Currently, LoRA adapters are only applied to the `query` and `value` projections. Extending them to `key` or `output` layers—or using multi-head LoRA—could capture more complex adaptation signals without a substantial increase in trainable parameters.
- **Learning Rate Scheduling:** While linear scheduling with warmup is effective, experimenting with cosine annealing or a one-cycle policy may yield smoother convergence and improved generalization. Cosine annealing in particular can better retain learning momentum during late training phases.
- **Regularization Strategies:** Incorporating label smoothing or R-Drop could mitigate overconfidence in predictions, which is common in transformer models. These techniques help in improving calibration and generalization on out-of-domain examples.
- **Longer Training with Early Stopping:** Increasing the number of epochs while incorporating early stopping based on validation accuracy or loss can allow for better performance without overfitting or unnecessary computation.
- **Advanced Token Augmentation:** Introducing data-level augmentation techniques like synonym replacement, back-translation, or mixup in the embedding space could enhance robustness to lexical variation, especially for low-resource or imbalanced datasets.
- **Quantization-Aware Training:** For deployment scenarios, applying quantization-aware training (QAT) could help reduce model size and inference latency without a major loss in accuracy.
- **LayerDrop or Stochastic Depth:** Incorporating dropout at the layer level (LayerDrop) during training might further regularize the model, especially when finetuning large pre-trained backbones with limited new data.
- **Hybrid Fine-Tuning Approach:** Explore a hybrid strategy combining LoRA with other PEFT methods like prefix-tuning or adapters. This can improve flexibility and allow targeted fine-tuning across specific transformer components.
- **Model Ensembling:** Averaging predictions from multiple PEFT variants (e.g., different seeds or slightly altered LoRA configurations) can often yield modest but consistent gains in classification performance.

These strategies aim to push the performance boundaries of efficient finetuning while keeping training costs low and preserving model generalization, particularly on tasks with limited labeled data.

## Conclusion

Our RoBERTa-based model, fine-tuned using Parameter-Efficient Fine-Tuning (PEFT) via LoRA, provides an effective and resource-efficient solution for multi-class text classification. Despite training less than 1% of the total parameters, the model achieves competitive performance on the AG News dataset. This validates the viability of lightweight adaptation techniques in scenarios where full fine-tuning is impractical due to resource or deployment constraints.

While the core architecture of RoBERTa remains unchanged, LoRA injects adaptability in a highly efficient manner. Our evaluation also illustrates how simplistic approaches (like vanilla fine-tuning) can be outperformed by structured parameter-efficient adaptations when coupled with thoughtful training strategies.

Future directions could include experimenting with expanded LoRA coverage, hybrid PEFT methods, and more diverse textual augmentation techniques. By continuing to explore this efficient adaptation landscape, we can push model performance further while keeping training and inference budgets in check.

## References

[1] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. In *arXiv preprint arXiv:1907.11692*, 2019.

[2] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *arXiv preprint arXiv:2106.09685*, 2021.

[3] Y. Zhu, X. Yang, Y. Wu, and W. Zhang. Parameter-Efficient Fine-Tuning with Layer Pruning on Free-Text Sequence-to-Sequence Modeling. In *arXiv preprint arXiv:2305.08285*, 2023.

[4] D. Hendrycks and K. Gimpel. Gaussian Error Linear Units (GELUs). In *arXiv preprint arXiv:1606.08415*, 2023.

[5] Y. Xu, L. Xie, X. Gu, X. Chen, H. Chang, H. Zhang, Z. Chen, X. Zhang, and Q. Tian. QA-LoRA: Quantization-Aware Low-Rank Adaptation of Large Language Models. In *arXiv preprint arXiv:2309.14717*, 2023.