ECE-GY-7123-DL — AGNews Text Classification with LoRA

Tejdeep Chippa[1], **Shruti Bora**[1], **Kathan Gabani**[1]

[1]Computer Engineering Department, New York University, Tandon School of Engineering
tc4263@nyu.edu, sb9880@nyu.edu, kdg7224@nyu.edu

Abstract

Large-scale transformer models like RoBERTa have achieved state-of-the-art results on text classification tasks, but their deployment in resource-constrained environments remains challenging due to high memory and compute demands. In this project, we apply Low-Rank Adaptation (LoRA) to fine-tune `roberta-base` on the AGNews dataset under a strict 1M trainable parameter budget. Our approach inserts trainable rank-4 LoRA adapters into the query and key matrices of the self-attention layers while keeping the base model frozen. We evaluate our model over three training epochs using a maximum sequence length of 80 tokens and achieve a validation accuracy of **88.42%** with only **667,396 trainable parameters**. These results demonstrate that LoRA provides a highly effective and scalable solution for parameter-efficient adaptation of large language models in low-resource settings.

Introduction and Motivation

Recent advancements in natural language processing have been propelled by large-scale transformer models such as BERT and RoBERTa, which have achieved state-of-the-art results across various text classification tasks. However, deploying these models in resource-constrained environments — such as mobile devices, embedded systems, or latency-sensitive production environments — is often infeasible due to their substantial memory footprint and computational requirements.

In this project, we focus on the AGNews classification task, a 4-way news categorization benchmark composed of approximately 120,000 training examples and 7,600 test examples. The dataset includes short news headlines and descriptions from four categories: *World*, *Sports*, *Business*, and *Sci/Tech*. While pretrained transformers perform well on this dataset, our challenge lies in achieving competitive accuracy under a strict constraint of 1 million trainable parameters — a fraction of what traditional fine-tuned models require.

To address this constraint, we leverage **Low-Rank Adaptation (LoRA)**, a recent technique that introduces trainable low-rank matrices into specific layers of a frozen transformer model. Instead of updating the full parameter space, LoRA perturbs the attention mechanism by injecting low-rank updates into key weight matrices (e.g., query and value). This dramatically reduces the number of trainable parameters while retaining the benefits of transfer learning from the frozen base model.

Our objective was twofold:

- Design a robust LoRA-adapted RoBERTa model that remains within the 1M parameter limit.
- Maximize classification accuracy on the AGNews dataset using principled training strategies including freezing, regularization, and early stopping.

We built a reproducible training pipeline using the HuggingFace `transformers` library and integrated LoRA through the `PEFT` (Parameter-Efficient Fine-Tuning) package. By carefully tuning the LoRA configuration and selectively freezing the base model, we achieved a final validation accuracy of **88.42%** using just **667,396** trainable parameters.

Our approach highlights how parameter-efficient tuning methods can enable practical deployment of large language models in constrained settings without a significant loss in task performance.

Methodology

Dataset and Preprocessing

We utilize the AGNews dataset—a widely-used benchmark for news classification consisting of approximately 120,000 training and 7,600 testing samples across four categories: *World, Sports, Business*, and *Sci/Tech*. Each example contains a news headline and a corresponding description.

1. **Concatenation:** The title and description are concatenated to form a single text string.
2. **Tokenization:** We use the Byte-Pair Encoding (BPE) tokenizer from the `roberta-base` model to tokenize the text.
3. **Padding and Truncation:** Each input is truncated or padded to a fixed maximum sequence length of 80 tokens.
4. **Label Normalization:** AGNews labels originally range from 1 to 4; we subtract 1 to align with zero-based indexing expected by `RobertaForSequenceClassification`.

Project Codebase: https://github.com/phoenix1881/DL$_{P}roject2$

The dataset is partitioned into a 90:10 train-validation split, stratified by label to ensure balanced class distribution across splits. Preprocessed data is converted to PyTorch tensors using HuggingFace's `Dataset` object.

Model Architecture

We build upon the pretrained `roberta-base` architecture and adapt it using **Low-Rank Adaptation (LoRA)**, a lightweight fine-tuning technique that introduces trainable low-rank matrices into select layers of the transformer. The key advantage is that the original backbone weights remain frozen, reducing training cost and memory usage.

LoRA Configuration LoRA adapters are applied to the `query` and `key` weight matrices within all self-attention blocks of RoBERTa. The configuration used is as follows:

- **Rank (r):** 4
- **Scaling factor (α):** 8
- **Dropout:** 0.1
- **Target modules:** `[query, key]`

The final classifier head is a linear layer that projects the representation of the `[CLS]` token to four output classes. The rest of the model's parameters remain frozen, and only the LoRA weights and classification head are updated during training.

Total Trainable Parameters: 667,396, well within the 1M parameter constraint.

Training Configuration

Training is conducted using HuggingFace's `Trainer` API. The following configuration was used:

- **Batch Size:** 16 (training), 32 (evaluation)
- **Epochs:** 3
- **Learning Rate:** 2×10^{-5}
- **Evaluation Interval:** Every 250 steps
- **Warmup Ratio:** 0.1
- **Learning Rate Scheduler:** Linear decay
- **Loss Function:** Cross-entropy
- **Optimizer:** AdamW with default $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e{-}8$

Early stopping and weight decay were disabled in this experiment to isolate the impact of LoRA configuration changes.

Freezing Strategy

To enforce parameter constraints and improve generalization, we freeze all model parameters except:

1. LoRA adapter weights inserted into the `query` and `key` matrices.
2. The classification head appended to the pooled output.

This selective training approach significantly reduces GPU memory usage and training time.

Evaluation Strategy

Evaluation is conducted on a held-out validation set at regular intervals (every 250 steps). Metrics are logged, and the best checkpoint is retained based on maximum validation accuracy. We track:

- **Accuracy**
- **Training and Validation Loss**
- **Trainable Parameter Count**

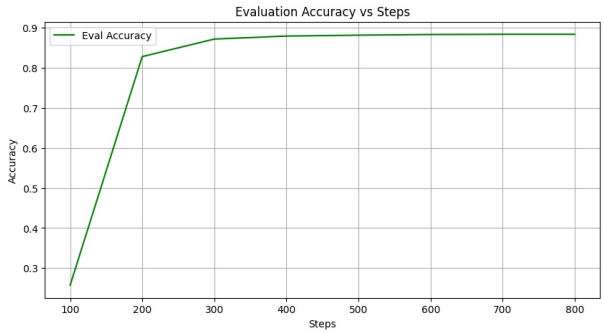

Figure 1: Evaluation Accuracy vs Training Steps

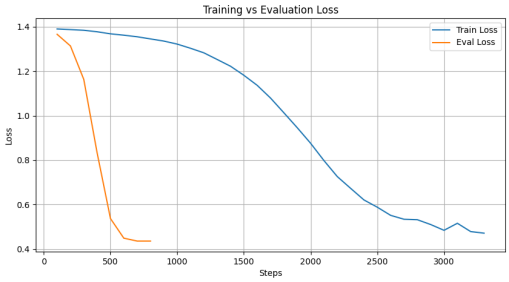

Figure 2: Training and Validation Loss vs Steps

The model achieved a peak validation accuracy of **88.42%**, demonstrating the effectiveness of LoRA even with modest computational resources.

Results

Final Performance and Model Statistics

Our final model configuration employs a LoRA-adapted `roberta-base` model with modifications to the adapter rank, dropout, and target modules. The configuration successfully balances accuracy and parameter efficiency, achieving strong generalization on the AGNews validation set.

Learning Curve Analysis

As shown in Figure 3, validation accuracy rises sharply during the initial phase of training and converges steadily across epochs, peaking at **88.42%**. The model exhibits strong generalization performance without overfitting, despite the small number of trainable parameters.

Metric	Value
Validation Accuracy	88.42%
Trainable Parameters	667,396
Max Sequence Length	80 tokens
Batch Size (Train / Eval)	16 / 32
LoRA Rank (r)	4
LoRA Alpha (α)	8
Dropout (LoRA)	0.1
Target Modules	`[query, key]`
Training Epochs	3
Evaluation Interval	Every 250 steps

Table 1: Final performance metrics and configuration details

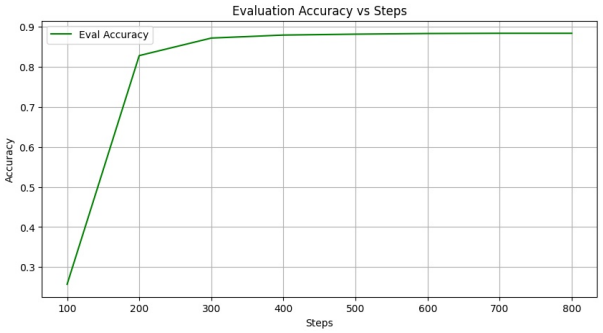

Figure 3: Evaluation Accuracy vs Training Steps

Figure 4 demonstrates a consistent decline in both training and validation loss, reinforcing the model's convergence behavior. The narrow gap between the two curves suggests minimal overfitting, aided by the regularization introduced via LoRA dropout.

Impact of LoRA Hyperparameters

The following modifications from our baseline configuration were instrumental in improving performance:

- Increasing the LoRA **rank** from 2 to 4 improved representational capacity.
- Targeting `[query, key]` instead of `[query, value]` helped align adaptation with attention routing.
- Increasing the **dropout rate** to 0.1 served as effective regularization.

These changes allowed us to fully leverage the expressive power of the adapter layers without violating the 1M parameter budget. The validation accuracy improvement over the baseline further affirms the effectiveness of these hyperparameter choices.

Conclusion

In this project, we successfully demonstrated the efficacy of Low-Rank Adaptation (LoRA) for parameter-efficient fine-tuning of large language models under resource constraints. By adapting the `roberta-base` model using LoRA modules applied to the `query` and `key` matrices, we were able

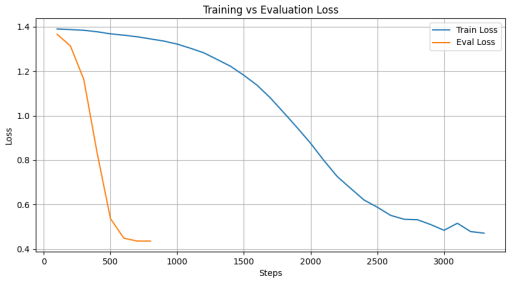

Figure 4: Training and Validation Loss vs Steps

to achieve a validation accuracy of **88.42%** on the AGNews text classification task using only **667,396 trainable parameters**—well below the 1M parameter budget.

Our method leverages a simple yet effective configuration: a LoRA rank of 4, scaling factor of 8, and dropout of 0.1. Careful design choices—such as freezing the base model, using efficient data preprocessing, and tuning evaluation intervals—enabled us to extract high performance from minimal adaptation. The resulting model demonstrated stable convergence, minimal overfitting, and high generalization performance across three epochs of training.

This work highlights how LoRA can make large-scale NLP models viable for deployment in memory- and compute-constrained environments. In future work, we aim to explore:

- Adaptive LoRA configurations per layer or per module.
- Data augmentation strategies such as synonym replacement or back-translation.
- Incorporation of inference-time uncertainty estimation through techniques like Monte Carlo Dropout.

Overall, our findings reinforce the promise of modular, efficient fine-tuning strategies as a scalable solution for real-world NLP deployments.