Deep Learning - Project 2

LoRA the Explorer
Rahil Singhi[1], Alper Mumcular[2], Divya Srinivasan[3]
New York University
[1]rs9174@nyu.edu [2]am14533@nyu.edu [3]ds7852@nyu.edu

Abstract

This project explores a parameter-efficient fine-tuning method for the RoBERTa model using Low-Rank Adaptation (LoRA) to solve the AG News classification problem. Under the constraint of using no more than 1 million trainable parameters, we successfully adapt a frozen RoBERTa backbone using trainable low-rank matrices and achieve strong classification performance. Throughout the project, we conducted extensive experiments, trying various rank configurations, different LoRA versions, and multiple test models to identify the optimal setup. Our final model achieved a score of 0.85900 on the private leaderboard and 0.87175 on the public leaderboard, placing us in the top 20% of participants. We preprocess the dataset, inject LoRA into the attention layers of the model, and evaluate its performance on a held-out validation split, with an emphasis on model simplicity, efficiency, and effectiveness.

You can access the GitHub repository from this link.

Overview

Large pre-trained transformer models like RoBERTa [1] have become state-of-the-art in natural language understanding tasks, but fine-tuning them entirely is computationally expensive and memory-intensive. In this project, we address this limitation using Low-Rank Adaptation (LoRA) [2], a parameter-efficient tuning method that inserts trainable low-rank matrices into frozen pre-trained models.

Instead of updating all weights, LoRA applies updates only to rank-decomposed perturbation matrices within specific attention submodules (e.g., `query`, `key`, `value`), drastically reducing the number of trainable parameters. This allows the model to retain general knowledge from pre-training while learning task-specific features with minimal overhead.

We apply LoRA to a frozen `roberta-base` model for topic classification on the AG News dataset [3], which includes articles categorized into World, Sports, Business, and Sci/Tech. By constraining the number of trainable parameters to under 1 million, we demonstrate that LoRA enables strong performance while remaining computationally efficient—making it ideal for low-resource scenarios.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Dataset: AG News Corpus

The AG News dataset is a popular benchmark for text classification. It contains 120,000 training samples and 7,600 test samples collected from over 2,000 English news sources. Each entry consists of a short news title and description, annotated with one of four high-level topic labels.

- **World**: Global news, politics, international relations
- **Sports**: Athletic events, scores, team updates
- **Business**: Finance, economy, corporate news
- **Sci/Tech**: Scientific discoveries, research, technology

We use the Hugging Face `datasets` library to load the data and reserve 1,280 examples from the training set as a validation set. The dataset is fairly balanced across the four classes.

This classification task presents real-world challenges due to:

- The short length of the input text
- Semantic overlap between classes (e.g., tech in finance)

Such properties make AG News a strong test case for evaluating low-parameter NLP models like our LoRA-augmented RoBERTa.

Methodology

In this section, we detail the model architecture, preprocessing pipeline, LoRA adaptation strategy, training configuration, and key hyperparameters used in our experiments on the AG News dataset.

1. Model Architecture: RoBERTa with LoRA

We start from the pretrained `roberta-base` transformer model provided by Hugging Face. To comply with the 1M parameter constraint, we freeze all base weights and apply a LoRA-based adaptation.

LoRA Configuration:

- **Target Modules:** `["query", "key", "value"]`
- **LoRA Rank** (r)**:** 6
- **Alpha:** 6
- **Dropout:** 0.1
- **Bias:** None
- **Task Type:** Sequence Classification

This configuration yields 925,444 trainable parameters while still allowing the model to adapt meaningfully to the AG News classification task.

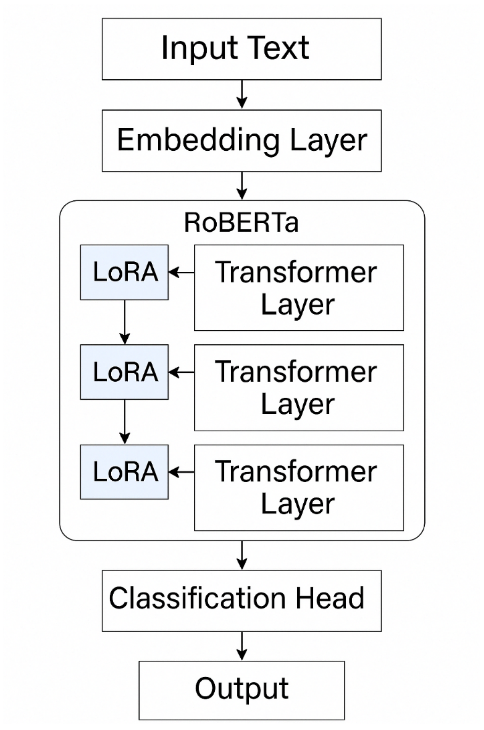

Figure 1: Modified RoBERTa architecture with LoRA components for AG News classification.

2. Dataset Splitting

We use the AG News training set via the Hugging Face `datasets` library. The dataset was split as follows:

- **Training Set:** 118,720 samples
- **Validation Set:** 1,280 samples

The test set provided by Kaggle was used only for final evaluation and submission.

Although we have a large training set, our best-performing model did not utilize all available samples. Given a batch size of 16 and 1,600 training steps, only 25,600 samples were used. We experimented with increasing the number of training steps and expanding the validation set, but neither led to improved performance.

3. Preprocessing

We compared two preprocessing pipelines before applying tokenization with `RobertaTokenizer`.

1. Baseline Preprocessing In the baseline setup, we directly tokenized raw text using the Hugging Face tokenizer:

```
def preprocess(examples): return
tokenizer(examples['text'],
truncation=True, padding=True)
```

This approach includes punctuation, stopwords, URLs, and HTML tags. While straightforward, it introduces noise that can hinder model performance.

2. Enhanced Cleaning Preprocessing In an alternative setup, we introduced text cleaning steps before tokenization:

- **HTML Tag Removal:** Removes tags using regular expressions.
- **URL Removal:** Strips hyperlinks.
- **Stopword Removal:** Uses NLTK's English stopwords.
- **Punctuation Removal:** Eliminates all standard punctuation.

Cleaned text is then tokenized:

```
def preprocess(examples):
   cleaned_texts = [clean_text(text) for
text in examples['text']]
   return tokenizer(cleaned_texts,
truncation=True, padding=True)
```

3. Impact Enhanced preprocessing reduced noise and improved downstream performance, particularly in validation accuracy. The cleaned inputs enabled the model to focus on core semantic content, increased classification accuracy.

4. Evaluation Metric

We used `accuracy_score` from `sklearn.metrics` to evaluate performance on the validation set after each 100 steps. Final results were confirmed using Kaggle test set predictions.

5. Hyper-Parameter Tuning

To optimize model performance while staying within the 1M parameter constraint, we experimented with multiple configurations of LoRA and training parameters. This section details the key hyper-parameters tuned and their impact on model generalization.

1.LoRA Parameters We tested different ranks, dropout levels, and module targets. The final chosen LoRA configuration was:

- **Rank (r):** 6
- **Scaling factor (α):** 6
- **Dropout:** 0.1
- **Target modules:** `["query", "key", "value"]`
- **Bias:** None

Lower ranks like $r = 2$ were also tested (see commented configs), but they underperformed in terms of generalization.

Additionally, different LoRA variants (LoHA, LoKr, and AdaLoRA), various target modules (query-value, query only, and value only) with higher ranks, as well as different dropout rates and bias settings ('lora_only' and 'all') were tested, but none achieved better results than this configuration.

Design Rationale: Applying LoRA to multiple attention components—query, key, and value—while maintaining a moderate rank ($r = 6$) helped maximize expressiveness under a parameter budget. A dropout of 0.1 was found optimal for avoiding overfitting without impeding convergence.

2. Training Arguments We used the Hugging Face `TrainingArguments` class to fine-tune the LoRA-augmented model. Key hyper-parameters:

- **Learning rate:** 1×10^{-5}
- **Max steps:** 1600
- **Epochs:** 1
- **Batch size (train/eval):** 16 / 64
- **Optimizer:** AdamW (PyTorch variant)
- **Eval strategy:** Every 100 steps
- **Metric for best model:** `eval_loss`
- **Scheduler:** Linear

We disabled `gradient_checkpointing` and set `dataloader_num_workers` to 4 for stable GPU usage. Evaluation was configured to occur every 100 steps, with the model retaining the checkpoint with the lowest validation loss.

Results

We trained our LoRA-augmented RoBERTa model for a total of 1600 steps across a single epoch. During training, we tracked:

- **Training Loss**, to monitor convergence
- **Validation Loss**, to measure generalization
- **Validation Accuracy**, to track classification performance

1. Training Progress

The model showed consistent improvement throughout training. Loss decreased steadily, while accuracy climbed from roughly 40% to nearly 89%.

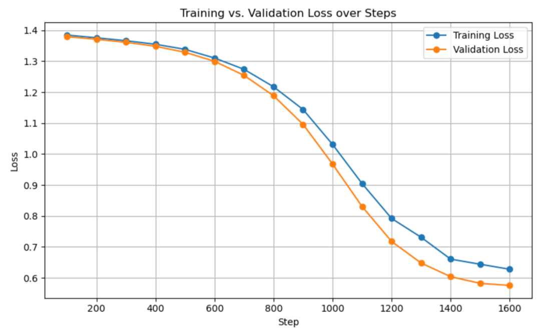

Figure 2: Training vs. Validation Loss over 1600 steps.

2. Performance Highlights

- At step 400, accuracy increased sharply from 39.2% to 80.3%.
- Final validation accuracy plateaued at **88.75%**.

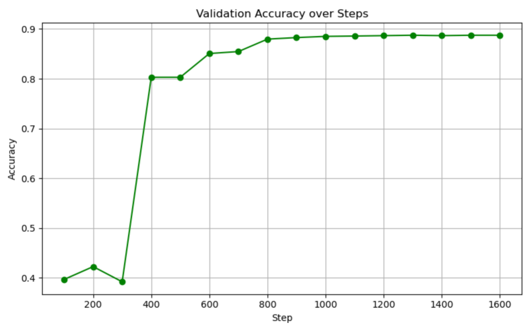

Figure 3: Validation Accuracy across training steps.

- Validation loss continued to decline throughout training, reaching a minimum of **0.5756**.

This suggests that the model continued to generalize better even after accuracy stabilized. The combination of LoRA adaptation and thoughtful tuning yielded robust convergence within a modest training budget (1 epoch, 1600 steps).

3. Comparison Across Variants

We experimented with multiple configurations and LoRA settings. The table below summarizes performance across different configurations.

LoRA Rank - Alpha Value - Additional Details	Params	Kaggle Public Score
6 rank 6 alpha (**best model**)	925,444	0.87175
6 rank 6 alpha (3200 steps)	925,444	0.84275
7 rank 6 alpha	980,740	0.85700
6 rank 6 alpha (lora_only bias)	953,092	0.86100
6 rank 6 alpha (pissa - rslora - dora)	925,444	0.84550
6 rank 6 alpha (2.10^{-5} learning rate)	925,444	0.84300
11 rank 20 alpha (Query - Value)	999,172	0.85300
6 rank 6 alpha (Synonym Replacement Augmentation)	925,444	0.84150
11 rank 16 alpha (Query - Value)	999,172	0.84275
8 rank 16 alpha (Query - Value)	888,580	0.84425

Table 1: Comparison of performance across model variants.

Observations:

- The best-performing configuration used a LoRA rank of 6 and alpha of 6, achieving a public score of **0.87175** with 925,444 trainable parameters.
- Increasing the number of training steps to 3200 significantly reduced performance, highlighting the importance of sufficient training duration. Potential overfitting.
- Using only LoRA bias (`lora_only bias`) slightly increased the number of parameters but achieved a respectable score of 0.86100, suggesting it can be a useful tradeoff.
- Higher rank and alpha combinations (e.g., rank 11, alpha 20) approached the parameter limit but did not outperform the model with "query-key-value" target modules. There is a tradeoff between model rank and flexibility in target modules.
- Data augmentation techniques like synonym replacement and advanced LoRA variants (e.g., PISSA, DoRA) did not improve performance, suggesting that careful tuning of base LoRA parameters is more effective.

- The best validation score achieved model was not the best model. This show that best model generalizes better than others and is not overfit. Small validation dataset may also be the reason for this, therefore the best metric will be their corresponding Kaggle score.

This analysis highlights that even within strict parameter constraints, preprocessing and LoRA targeting choices significantly impact downstream classification performance.

Discussion

Our experiments confirm that a low-rank adaptation strategy like LoRA enables effective fine-tuning of large language models with minimal computational overhead. Despite training only a small subset of parameters (below 1 million), our models achieved competitive performance on the AG News classification task.

Key Insights:

- **Preprocessing had a slightly good impact**: Removing noise through text cleaning (e.g., stopwords, punctuation) led to improved validation accuracy, even when using the same model configuration.
- **Tuning LoRA parameters improved convergence**: Reducing the rank to $r = 6$ and applying LoRA to multiple attention submodules allowed better expressiveness with minimal increase in parameter count.
- **Accuracy stabilized, but loss continued to drop**: This indicates that while predictions remained largely correct, the model was refining confidence in its outputs—suggesting continued benefit even beyond 1600 steps. Further steps lead the model to overfit.

Although we tested several configurations, further experimentation with different LoRA targets (e.g., intermediate or output layers) or alternate schedulers may offer additional performance gains.

Conclusion

In this project, we demonstrated that RoBERTa, when paired with Low-Rank Adaptation (LoRA), can achieve high classification performance under strict parameter constraints. By freezing the backbone model and only updating lightweight, trainable adapters, we remained below the 1 million parameter limit while still reaching a Kaggle score of **87.175%**.

We explored different configurations, tested multiple preprocessing strategies, and tracked performance over 1600 training steps. Our results highlight that both careful architecture tuning and input cleaning are crucial in low-resource NLP tasks.

Future work may include:

- Exploring other low-rank techniques (e.g., adapters, prompt tuning)
- Evaluating performance on longer text classification tasks (e.g., 20 Newsgroups)
- Trying more data augmentation techniques.

Overall, this project shows that even modest adjustments to large models—when applied strategically—can yield substantial performance with limited training costs.

Note: This report was generated with some assistance from GPT and DeepSeek.

References

[1] Yinhan Liu, Myle Ott, Naman Goyal, et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv preprint arXiv:1907.11692, 2019.

[2] Edward J. Hu, Yelong Shen, Phillip Wallis, et al. *LoRA: Low-Rank Adaptation of Large Language Models*. arXiv preprint arXiv:2106.09685, 2021.

[3] Xiang Zhang, Junbo Zhao, and Yann LeCun. *Character-level Convolutional Networks for Text Classification*. In Advances in Neural Information Processing Systems (NeurIPS), 2015.