Project 2 - Parameter-Efficient Fine-Tuning of RoBERTa for AG News Classification Using LoRA

Abhishek Adinarayanappa (aa12037), Harsha Mupparaju (sm12754), Nived Damodaran (nd2746)

GitHub Repository Link: https://github.com/Abhi270600/LLM-finetuning-using-LoRA

Abstract

This project explores the application of Low-Rank Adaptation (LoRA) for parameter-efficient fine-tuning of a pretrained RoBERTa-base model on the AG News text classification dataset. LoRA enables fine-tuning large language models (LLMs) by injecting trainable low-rank matrices into their weight structures while keeping the majority of the pretrained weights frozen. This dramatically reduces the number of trainable parameters and training costs, making it ideal for resource-constrained environments.

We fine-tuned RoBERTa-base using LoRA to classify news headlines into four categories: World, Sports, Business, and Sci/Tech. Our approach emphasizes model efficiency and scalability while maintaining competitive accuracy. The final model achieved a test accuracy of 84.825%, validating the effectiveness of LoRA in practical NLP scenarios. Further improvements could be achieved by optimizing hyperparameters or experimenting with different LoRA configurations.

Methodologies

Our approach consists of fine-tuning the RoBERTa-base model using LoRA adapters while evaluating model performance on the AG News dataset.

Dataset

The AG News dataset is a widely-used benchmark for text classification, consisting of over 120,000 short news articles categorized into four balanced classes: **World**, **Sports**, **Business**, and **Sci/Tech**. Each data sample includes a **title** and a **description**, which together provide a concise summary of a news story.

We leveraged the HuggingFace `datasets` library to download and preprocess the dataset efficiently. The original dataset is split into:

- **Training set:** 120,000 samples
- **Test set:** 7,600 samples

To monitor validation performance during training, we further split the original training set into:

- **Fine-tuning set:** 119,308 samples
- **Validation set:** 640 samples (randomly selected for quick evaluation)

Preprocessing

- **Text cleaning:** Lowercasing, whitespace normalization, and punctuation handling were performed for consistency.
- **Tokenization:** Used the RoBERTa tokenizer with truncation and padding to a fixed maximum sequence.
- **Class distribution:** The dataset is relatively balanced across the four classes, with each category containing approximately 30,000 samples in the training set.

This preprocessing ensures that the input format aligns with the expectations of the RoBERTa model while preserving the semantic structure of the news data. The relatively short and structured nature of AG News samples makes it suitable for benchmarking low-parameter fine-tuning strategies such as LoRA.

LoRA Integration and Model Architecture

LoRA (Low-Rank Adaptation) is a parameter-efficient fine-tuning method designed to adapt large pre-trained language models (like GPT or BERT) with minimal computational overhead. Instead of fine-tuning all the model's parameters, LoRA freezes the original weights and introduces small, trainable low-rank matrices to approximate weight updates.

Key Idea of LoRA

LoRA is based on the observation that weight updates during fine-tuning often have a low **intrinsic rank**—meaning they can be decomposed into smaller matrices without losing much information. Instead of modifying the original large weight matrix $W \in R^{d \times k}$, LoRA represents the update ΔW as a product of two low-rank matrices:

$$\Delta W = B \cdot A$$

where:

- $B \in R^{d \times r}$ (low-rank projection down)
- $A \in R^{r \times k}$ (low-rank projection back up)
- $r \ll \min(d, k)$ (the rank is much smaller than original dimensions)

During fine-tuning, only A and B are trained, while the original W remains frozen.

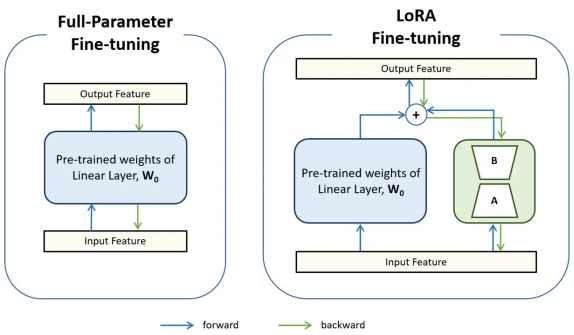

Figure 1: LoRA Architecture

Techniques Used

Data Preprocessing To evaluate the impact of text preprocessing on model performance, stemming and lemmatization were applied to the input data. Stemming is a rule-based technique that reduces words to their root forms by trimming word endings, often producing non-dictionary terms. Lemmatization, in contrast, maps words to their base dictionary form using linguistic rules, offering higher semantic accuracy. These techniques are commonly beneficial in tasks like sentiment analysis; however, in this experiment, neither stemming nor lemmatization led to a measurable improvement in model accuracy.

Student-Teacher Distillation Knowledge distillation is a model compression technique where a smaller student model learns to mimic the behavior of a larger teacher model by matching its output distributions. This allows the student to achieve competitive performance with reduced computational cost and fewer parameters. Here we used LoRA-parameterized RoBERTa (890k parameters) as the student model and roberta-base as the teacher model. The loss function used was a combination of Cross entropy loss and KL Divergence loss, given by

$$\text{Total Loss} = \alpha \times \text{KL Divergence Loss} + (1-\alpha) \times \text{Cross Entropy Loss}$$

Key Hyperparameters used are:

- Alpha(distillation loss weight) : 0.8
- Temperature: 2
- LoRA Rank: 16
- Alpha(LoRA): 32

Contextual Data Augmentation To improve model robustness and diversity in training data, contextual word-level augmentation was performed using the Hugging Face masked language model pipeline.Each input sentence had a random word masked and replaced with contextually relevant predictions from the roberta-base model. This increased the training dataset from 120k samples to 360k samples, generating 2 more samples for each original sample.

Final Training Strategy and Hyperparameters

The training was conducted using PyTorch and HuggingFace's Trainer API with the following key hyperparameters:

- Optimizer: Adam
- Learning Rate: 2e-4
- Learning Rate Strategy: Linear with warmup
- Batch Size: 32
- Epochs : 1
- LoRA ranks: $r \in \{2, 4, 8, 16\}$
- Alpha parameter: $\alpha = 2r$
- Target modules: query, key, value and other projection layers in the `roberta-base` model architecture

The classification head used a single linear layer mapping the RoBERTa hidden states to four output classes.

Evaluation Metrics

Model performance was evaluated using:

- Accuracy
- Precision
- Recall
- F-1 Score

We monitored validation accuracy during training and selected the best model checkpoint based on it.

Results

The final model which achieved the best results had the parameters: $r = 9$, $\alpha = 32$, lora_dropout $= 0.1$, target_modules = [query ,value] (Accuracies rounded to 2 decimals):

- Test Accuracy: **84.83%**
- Evaluation Accuracy: **93.12%**
- Evaluation Precision: **93.13%**
- Evaluation Recall: **93.12%**
- Evaluation F-1: **93.12%**
- Trainable Parameters: **925,444(0.7370%)** Total parameters: **125,574,152**

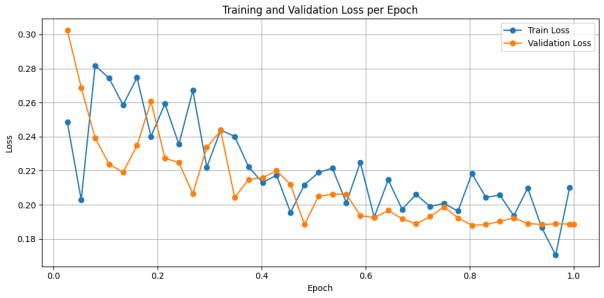

Figure 2: Training and Validation Loss

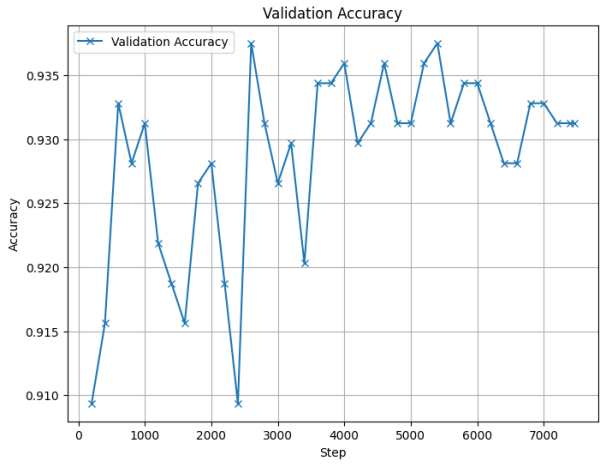

Figure 3: Validation Accuracy

Observations

- LoRA-enabled fine-tuning was significantly faster and less memory-intensive than full fine-tuning.
- With proper adapter placement, accuracy reached competitive levels with full-model fine-tuning.
- Minor class imbalance was observed, but LoRA still generalized well.
- An accuracy of 83.43% was achieved using preprocessing techniques such as data cleaning and lemmatization. The Student–Teacher Distillation approach yielded a higher accuracy of 84.25%, When applying contextual data augmentation, the model achieved an accuracy of 83.08%. Surprisingly, none of these techniques outperformed the accuracy achieved through naive LoRA fine-tuning

Potential Improvements

- **LoRA Parameter Exploration:** Further grid search on LoRA rank, α values, and dropout rates may yield better trade-offs between parameter count and accuracy. Additionally, exploring layer-wise rank customization could offer improvements.
- **Alternative PEFT Techniques:** Combining LoRA with other parameter-efficient fine-tuning (PEFT) techniques such as Prefix Tuning, AdapterFusion, or BitFit may enhance performance while preserving model efficiency.
- **Data Augmentation Diversity:** While contextual augmentation helped increase training data diversity, applying synonym replacement, back-translation, or paraphrasing methods may introduce even more robust variations.
- **Handling Class Imbalance:** Employing techniques such as weighted loss functions or oversampling underrepresented classes may mitigate mild class imbalance and boost per-class performance.
- **Extended Contextual Inputs:** Incorporating additional metadata (e.g., publication date or article source) or increasing the maximum input token length could provide richer contextual cues and potentially improve classification accuracy.
- **Model Ensembling:** Building ensembles of LoRA-tuned models with varied ranks or initialization seeds could lead to improved robustness and predictive performance on the test set.

References

https://huggingface.co/docs/peft
https://arxiv.org/abs/2106.09685 (LoRA paper)
https://huggingface.co/docs/transformers
https://www.kaggle.com/datasets/amananandrai/ag-news-classification-dataset