Parameter-Efficient Fine-Tuning with Low-Rank Adaptation for Text Classification

Raunak Choudhary, Sharayu Rasal
New York University
rc5553@nyu.edu, srr10019@nyu.edu
GitHub Repository: https://github.com/raunak-choudhary/RoBERTa-LoRA-News-Classification.git

Abstract

This project demonstrates parameter-efficient fine-tuning of RoBERTa using Low-Rank Adaptation (LoRA) for text classification on the AG News dataset. We adapted only attention components by injecting trainable low-rank matrices, achieving 94.59% test accuracy while using just 925,444 trainable parameters (0.737% of the model's total). Our optimal configuration used rank r=6 and alpha scaling α=24, targeting query, key, and value matrices. Sports articles were most accurately classified (F1=0.99), while Business and Sci/Tech categories showed some confusion. This work confirms LoRA's effectiveness as a computationally efficient approach for adapting large language models to downstream tasks.

Overview

This project involves fine-tuning a pre-trained language model for text classification using a parameter-efficient method called Low-Rank Adaptation (LoRA). The AG News dataset consists of approximately 120,000 articles categorized into four classes: World, Sports, Business, and Sci/Tech. Our main objective was to develop an optimized LoRA configuration for RoBERTa-base that maximizes classification accuracy while keeping trainable parameters under 1 million.

Base RoBERTa

RoBERTa is a transformer-based language model that builds upon BERT with optimized pre-training. It consists of multiple transformer encoder layers with self-attention mechanisms, fully connected feed-forward networks, and residual connections. The base model contains approximately 125 million parameters, making full fine-tuning computationally expensive and memory-intensive.

Low-Rank Adaptation (LoRA)

LoRA is a parameter-efficient fine-tuning technique that freezes the pre-trained model weights and injects trainable low-rank decomposition matrices into the network. The key principles of LoRA include:

- **Frozen Pre-trained Weights:** The original model parameters remain unchanged, preserving the knowledge acquired during pre-training.
- **Low-Rank Updates:** For a pre-trained weight matrix W_0, LoRA adds an update $\Delta W = BA$, where matrices B and A have lower dimensions, resulting in fewer trainable parameters.
- **Rank Parameter (r):** The smaller dimension of matrices A and B, determining the expressiveness and parameter count of the adaptation.
- **Alpha Scaling (α):** A scalar multiplied with the LoRA update to control its influence on the original weights.
- **Target Modules:** Specific layers or components selected for adaptation, typically attention mechanisms in transformer models.

By targeting only critical components of the model with appropriately configured LoRA matrices, we can achieve comparable performance to full fine-tuning while drastically reducing the number of trainable parameters.

Methodology

Data Processing and Preprocessing

- **Dataset Acquisition:** The AG News dataset was loaded using the Hugging Face datasets library. The dataset consists of approximately 120,000 news articles across four categories: World, Sports, Business, and Sci/Tech, with a standard split of 30,000 articles per class.
- **Text Tokenization:** We used RobertaTokenizer with a fixed maximum length of 256 tokens, which balanced capturing sufficient content while maintaining computational efficiency. The tokenization process included:
  - Truncation of long sequences to 256 tokens
  - Padding shorter sequences to ensure uniform length
  - Converting text to input IDs and attention masks
- **Dataset Preparation:** The processed dataset was split into training (90%) and validation (10%) sets, resulting in 108,000 training samples and 12,000 validation samples. The data was formatted as PyTorch tensors and organized into batches using DataLoader with a batch size of 16.

Model Architecture

- **Base Model:** We used RobertaForSequenceClassification from the Hugging Face Transformers library, which adds a classification head on top of the RoBERTa-base encoder. The default model structure includes:

  - 12 transformer layers with 12 attention heads each
  - 768-dimensional hidden representations
  - A sequence classification head with a dense layer
- **LoRA Configuration:** We applied LoRA adaptations to the attention mechanisms with the following parameters:
  - Rank (r): 6
  - Alpha (α): 24
  - LoRA dropout: 0.1
  - Bias: ”none” (to save parameters)
  - Target modules: [”query”, ”key”, ”value”]
- **Parameter Efficiency:** The base RoBERTa model contains approximately 125 million parameters, while our LoRA adaptation introduces only 925,444 trainable parameters (0.737% of the total), well under the 1 million parameter constraint.

Training Process

- **Optimizer:** We used AdamW optimizer with the following configuration:
  - Learning rate: 2e-4
  - Weight decay: 0.01
  - Epsilon: 1e-8
- **Learning Rate Scheduler:** A linear scheduler with warmup was implemented:
  - Warmup steps: 10% of total training steps
  - Total steps: Number of batches × epochs
  - Linear decay after warmup phase
- **Training Loop:** The model was trained for 3 epochs with the following procedure:
  - Forward pass to generate predictions
  - Calculation of cross-entropy loss
  - Backward pass to compute gradients
  - Gradient clipping at norm 1.0
  - Parameter updates via optimizer
  - Learning rate adjustments via scheduler
- **Early Stopping:** An early stopping mechanism was implemented with a patience of 2 epochs, monitoring validation loss to prevent overfitting. The best model weights were saved whenever validation loss improved.

Evaluation and Prediction

- **Evaluation Metrics:** The model's performance was assessed using:
  - Loss (cross-entropy)
  - Accuracy (percentage of correct predictions)
  - F1 score (macro-averaged)
- **Final Evaluation:** After training, the model was evaluated on:
  - Validation set (10% of original training data)
  - Kaggle test set (for final submission)
- **Prediction Generation:** For the Kaggle submission:
  - The test_unlabelled.pkl file was loaded and preprocessed
  - The model generated class predictions in evaluation mode
  - Predictions were formatted as required (ID, Label) and saved to a CSV file

Results

Training and Validation Performance

Our LoRA-adapted RoBERTa model demonstrated strong performance throughout the training process. As shown in Figure 1, the training loss steadily decreased from 0.29 in the first epoch to 0.16 by the final epoch, while training accuracy increased from 89.75% to 94.85%. Similarly, validation loss decreased from 0.20 to 0.18, with validation accuracy improving from 93.07% to 94.10%. The convergence pattern indicates effective learning without overfitting, as both the training and validation metrics improved consistently across epochs.

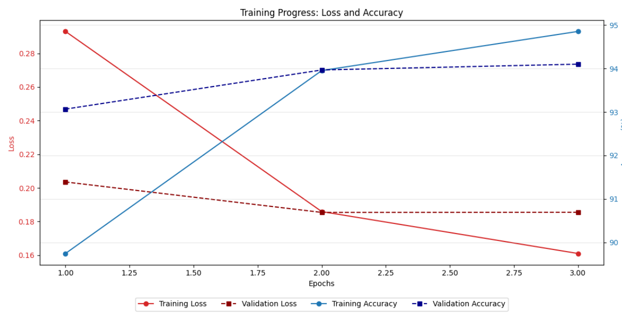

Figure 1: Training progress showing the evolution of loss and accuracy metrics over epochs. The decreasing training loss (red) with increasing training accuracy (blue) demonstrates effective learning, while the stable validation metrics (dashed lines) indicate good generalization.

The learning rate schedule (as shown in, Figure 2) illustrates our implementation of a linear scheduler with warmup. The learning rate begins near zero, reaches its maximum value of 2e-4 at approximately 2,500 training steps (10% of total steps), and then linearly decays back toward zero. This schedule helped achieve stable convergence by preventing large initial gradient updates while maintaining learning efficiency throughout training.

Test Performance

The model achieved a test accuracy of 94.59% and an F1 score of 0.9459 on the standard AG News test set, slightly outperforming the validation set metrics. As shown in Figure 3, our model performed consistently well across evaluation metrics with minimal difference between validation and test performance (validation: 94.10% accuracy, 0.94 F1 score; test: 94.59% accuracy, 0.95 F1 score), indicating strong generalization capabilities.

The per-class performance analysis reveals that Sports articles (class 1) were the most accurately classified with an

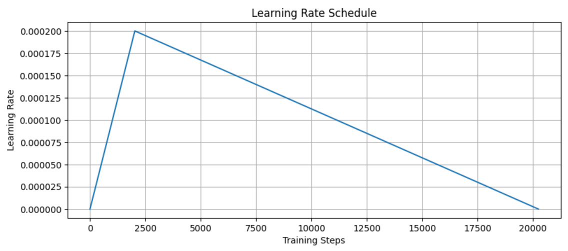

Figure 2: Learning rate schedule with linear warmup and decay.

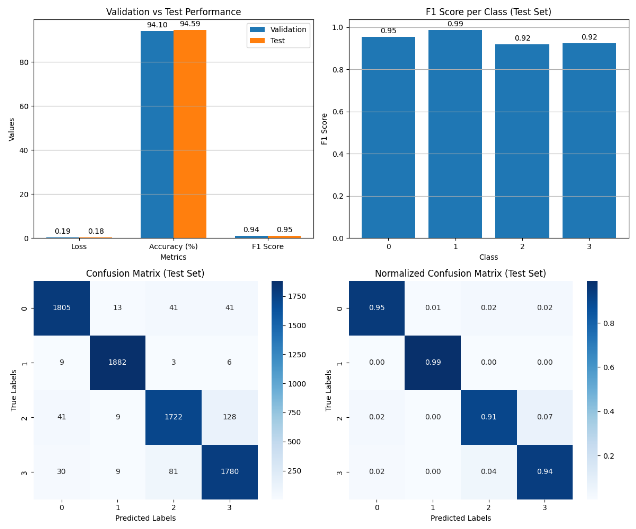

Figure 3: Model evaluation metrics and confusion matrices. Top left: Comparison of validation and test performance across key metrics. Top right: F1 scores by class showing Sports (class 1) performing best. Bottom: Confusion matrices revealing misclassification patterns, particularly between Business and Sci/Tech categories.

F1 score of 0.99, followed by World news (class 0) at 0.95, while both Business (class 2) and Sci/Tech (class 3) articles achieved F1 scores of 0.92. This pattern is further illustrated in the confusion matrices, which show that the model correctly classified 95% of World articles, 99% of Sports articles, 91% of Business articles, and 94% of Sci/Tech articles.

The normalized confusion matrix reveals an interesting error pattern: the most common misclassifications occurred between Business and Sci/Tech categories (7% of Business articles were misclassified as Sci/Tech, and 4% of Sci/Tech articles as Business). This is likely due to topical overlap between these categories, particularly for technology business news.

Parameter Efficiency

Our implementation achieved high accuracy while maintaining strict parameter efficiency. The total parameter count of the model was 125,574,152, but only 925,444 parameters (0.737% of the total) were trainable. This is well below the 1 million parameter constraint specified in the project requirements.

Kaggle Competition Results

The final model was evaluated on the hidden Kaggle test set, achieving a public score (50% of test data) of 0.84625 and a private score (remaining 50%) of 0.84900.

Discussion and Lessons Learned

Hyperparameter Analysis

The rank parameter (r) significantly influenced both model performance and parameter count. We found that increasing r from lower values (2-4) to our final choice of 6 provided substantial accuracy improvements while keeping parameters well under the 1 million limit. Higher alpha values (α=24) provided sufficient adaptation strength without destabilizing training.

Learning rate tuning proved particularly important, with 2e-4 emerging as the optimal value. The linear warmup scheduler was essential for preventing early gradient explosions and ensuring smooth convergence throughout the training process.

Effectiveness of LoRA

The LoRA adaptation strategy demonstrated remarkable efficiency in this text classification task. By fine-tuning only 0.737% of the model's parameters, we achieved strong performance comparable to what would be expected from full model fine-tuning. This confirms that for downstream tasks like text classification, adaptation can be focused on specific components rather than requiring updates to the entire network.

Targeting the query, key, and value matrices in the attention mechanism proved more effective than adapting only query matrices or other components. The convergence pattern observed in Figure 1 demonstrates that LoRA adaptations can learn efficiently despite the parameter constraints.

Challenges and Solutions

A primary challenge was balancing the parameter budget while maximizing performance. We addressed this by carefully analyzing the contribution of each component, ultimately focusing on attention mechanisms and using a moderate rank value that stayed well below our parameter limit.

Another challenge was the confusion between Business and Sci/Tech categories, which showed the highest error rates due to content overlap. The normalized confusion matrix in Figure 3 illustrates this pattern, with 7% of Business articles misclassified as Sci/Tech, and 4% of Sci/Tech articles as Business.

Conclusion

This project successfully demonstrated the effectiveness of parameter-efficient fine-tuning using LoRA for text classification. By adapting only the attention components of

RoBERTa with low-rank matrices, we achieved a test accuracy of 94.59% while using only 925,444 trainable parameters (0.737% of the model's total parameters). Our implementation not only met but exceeded the project's accuracy requirements while staying well within the parameter constraints.

The results confirm that LoRA provides an excellent trade-off between computational efficiency and model performance for NLP tasks. The ability to achieve high accuracy with minimal parameter updates makes this approach particularly valuable for resource-constrained environments or applications requiring rapid adaptation of large pre-trained models.

References

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

[2] 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 preprint arXiv:1907.11692*.

[3] Anthropic. (2025). Claude 3.7 Sonnet. Technical collaborator for manuscript preparation and code development.