Parameter-Efficient Fine-Tuning of RoBERTa on AGNEWS using LoRA

Dhrity Kachhwaha (dk5144), Sanyukta Tuti (st5442), Shivangi Raj (sr7731)
GitHub Repository: Deep Learning Project 2

Abstract

In this project, we fine-tuned a pre-trained RoBERTa-base model on the AG News dataset using Low-Rank Adaptation (LoRA) to perform four-way news classification. Leveraging parameter-efficient fine-tuning, our approach achieved a final test accuracy of *84.77

Introduction

Text classification remains a core task in natural language processing, with applications ranging from sentiment analysis to topic categorization. The AG News dataset comprises 120,000 training and 7,600 test samples across four categories (World, Sports, Business, Sci/Tech). Our goal was to leverage a transformer-based architecture with minimal trainable parameters to balance performance and resource efficiency.

Key objectives:

- Utilize a pre-trained `RoBERTa-base` model for strong language representations.
- Apply LoRA to drastically reduce the number of trainable parameters.
- Attain competitive accuracy within Kaggle competition constraints (no external data, <5M tuned parameters).

Methodology

Dataset and Preprocessing

1. **Data Acquisition:** Downloaded the `deep-learning-spring-2025-project-2` competition data via the Kaggle API and unzipped to extract `test_unlabelled.pkl` and related files.
2. **Training Split:** Used the Hugging Face `load_dataset` utility to load the AG News training split; created an internal 119,360/640 train/eval split for validation.
3. **Tokenization & Padding:** Employed `RobertaTokenizer` with truncation and padding to a fixed maximum length, ensuring uniform input sizes for batching.
4. **Data Collation:** Used `DataCollatorWithPadding` for dynamic padding at batch time, improving computational efficiency.

Model Architecture and LoRA Design

Base Model: `RoBERTa-base` (125M parameters) for robust contextual embeddings.

- **Pros:** State-of-the-art performance on many text tasks; well supported by Hugging Face.
- **Cons:** Large parameter count can be costly to fine-tune fully.

LoRA Configuration:

- Low-rank adaptation ($r = 11$, $\alpha = 8$, dropout $= 0.05$) applied to query and key projection matrices.
- Trainable parameters reduced to $\sim$1M (<1% of full model).
- **Pros:** Significant parameter efficiency; faster convergence and lower memory footprint.
- **Cons:** May limit expressiveness compared to full fine-tuning; requires careful hyperparameter tuning for rank and dropout.

PEFT Integration: Wrapped the RoBERTa model with the PEFT LoRA wrapper via `get_peft_model`; printed and verified trainable parameter counts.

Training Setup and Inference

- **Framework:** Leveraged Hugging Face `Trainer` for streamlined training and evaluation loops.
- **Hyperparameters:**
  - Learning rate: 9×10^{-4}
  - Epochs: 2
  - Batch sizes: 16 (train), 64 (eval)
  - Optimizer: `adamw_hf`
  - Early stopping patience: 10 evaluation steps
- **Hardware:** Preferred Apple MPS on macOS, fallback to CUDA/CPU.
- **Metrics:** Computed accuracy with `sklearn` metric; selected the best model via lowest eval loss.

- **Inference:** Ran batched inference on `test_unlabelled.pkl` and saved predictions to `results/inference_output.csv` for Kaggle submission.

Evaluation was performed at step intervals with `load_best_model_at_end=True`. Accuracy was the primary metric, measured using Hugging Face's `evaluate` library.

Design Reflections and Lessons Learned

- Hyperparameter Sensitivity: Our experiments showed that learning rates above 1e-3 led to unstable convergence, while lower rates improved stability at the cost of slower training.
- Model Efficiency vs. Expressiveness: LoRA adapters greatly reduced trainable parameters (~1M), but we observed a slight performance drop compared to full fine-tuning, suggesting a trade-off between efficiency and representational capacity.
- Validation/Test Gap: The discrepancy between high validation accuracy (93.75%) and private test performance (84.77%) highlighted the need for cross-validation or robust hold-out strategies to estimate generalization more reliably.
- Hardware Considerations*: Leveraging Apple MPS accelerated prototyping, but limited GPU support for bitsandbytes restricted the use of 8-bit optimizers, impacting memory efficiency.

  Results —

Split	**Accuracy**	**Loss**
Validation	93.75%	0.2012
Public Test (Kaggle)	85.90%	—
Private Test (Kaggle)	84.77%	—

Table 1: Model performance on different data splits

- The internal validation accuracy peaked at **93.75%**, indicating strong in-domain performance.
- The final Kaggle submission achieved **84.77%**, reflecting some domain shift between our validation split and the private test set.

Reproducibility

To reproduce these results:

- **Environment:**
  - Python 3.12
  - `transformers==4.46.3`, `datasets==3.5.0`, `peft==0.15.1`, `bitsandbytes==0.42.0`, `evaluate==0.4.3`
  - Hardware: macOS with MPS or Linux/Windows with CUDA support.
- **Code & Data:**
  - Kaggle API download command: `kaggle competitions download -c deep-learning-spring-2025-project-2`
  - Notebook: `Deep-Learning-Project-2.ipynb` (config sections, training, inference)
- **Random Seeds:**
  - Set `seed=42` during the `train_test_split` call.
- **Dependencies:**
  - Shared via `pip install transformers datasets accelerate peft trl bitsandbytes evaluate` commands at the top of the notebook.
- **Submission:**
  - Ensure `results/inference_output.csv` follows the required format (`ID,Label`).

Lessons Learned

- Parameter-Efficient Tuning: LoRA provided an excellent trade-off between resource usage and model performance, though rank selection requires careful tuning.
- Validation/Test Gap: High validation accuracy did not fully translate to the private test set, highlighting the need for cross-validation or robust hold-out strategies.
- Hyperparameter Sensitivity: Learning rate and early stopping patience significantly impacted convergence; smaller rates reduced oscillation.
- Hardware Considerations: Running on Apple MPS offered speedups for prototyping, but GPU support for bitsandbytes was lacking in our environment.
- Future Directions: Ensemble of LoRA adapters, longer training schedules, or domain-specific pre-training could further close the validation/test performance gap.

References

1. Hugging Face Transformers: https://huggingface.co/transformers/
2. AGNEWS Dataset: https://huggingface.co/datasets/ag_news
3. LoRA Paper: https://arxiv.org/abs/2106.09685
4. PEFT Library: https://github.com/huggingface/peft