Deep Learning Spring 2025: LoRA-Augmented RoBERTa for AGNews Text Classification
Kaggle challenge 2: Deep Learning SP25

Sarang Pinakin Kadakia[1], Rujuta Amit Joshi[2], Vishwajeet Kulkarni[3]

[1]New York University
[2] New York University
[3] New York University

sk11634@nyu.edu, rj2719@nyu.edu, vk2630@nyu.edu

GitHub Project Repository: Github Link

Abstract

In this project, we explore parameter-efficient fine-tuning (PEFT) techniques using Low-Rank Adaptation (LoRA) on RoBERTa, a transformer-based model, to perform efficient text classification on the AG News dataset. We aim to achieve optimal classification performance while significantly reducing the computational cost and memory requirements. Our implementation leverages Quantization (4-bit with BitsAndBytes), LoRA for selective parameter updating, and Optuna for hyperparameter optimization. Through systematic Optuna-based hyperparameter tuning, including learning rate, LoRA configurations (rank, alpha, dropout), and other transformer-specific parameters, we achieve competitive accuracy exceeding 93% on the evaluation set. Additionally, we emphasize maintaining the number of trainable parameters below 1 million to ensure computational efficiency. Our experimental results demonstrate that LoRA-enabled RoBERTa models, combined with strategic quantization and hyperparameter tuning, provide a robust yet resource-efficient solution for large-scale text classification tasks.

Introduction

In this project, we focus on solving a news categorization task using the AG News dataset, which consists of news headlines and descriptions labeled across four distinct classes. The objective is to build an efficient and high-performing deep learning pipeline to classify input news text into one of these categories. Leveraging the transformer-based RoBERTa model as our backbone, we incorporate Parameter-Efficient Fine-Tuning (PEFT) techniques, specifically Low-Rank Adaptation (LoRA), to reduce trainable parameters while maintaining high accuracy. This allows us to fine-tune large models on limited computational resources without compromising performance. Additionally, Optuna-based hyperparameter tuning was used to optimize key training configurations, enabling improved generalization to unseen test samples.

1.1. Motivation and Challenges

Despite being a four-class classification task, AG News presents multiple practical and modeling challenges:

- **Domain-Specific Semantics:** News headlines often contain concise, context-rich phrases that require nuanced understanding. Simple models struggle to capture the subtle semantic differences between classes like “World” and “Business.”
- **Low Resource Fine-Tuning:** Due to hardware constraints, we were restricted to fewer trainable parameters. This limited the use of full fine-tuning on large transformers like RoBERTa and motivated the adoption of parameter-efficient methods such as LoRA.
- **Generalization to Real-World Test Set:** The Kaggle test set contained slightly noisier and more varied samples compared to the validation set, highlighting the challenge of overfitting curated benchmarks and requiring careful hyperparameter optimization.

1.2. Residual Networks

Hu et al. [1] introduced LoRA (Low-Rank Adaptation) as a parameter-efficient fine-tuning method for large pre-trained models. Instead of updating all model weights, LoRA injects trainable rank-decomposed matrices into the attention layers (e.g., query, key, value projections) while freezing the original weights. This drastically reduces the number of trainable parameters while maintaining performance. By learning additive weight adjustments in a low-dimensional space, LoRA enables efficient adaptation under compute constraints, making it ideal for resource-limited settings like our AG News classification task.

1.3. Custom Test Dataset Context

A distinguishing aspect of this project was the requirement to evaluate performance on a custom, unseen test dataset provided via Kaggle. Unlike the original AG News dataset (120,000 labeled samples across four categories), the test set shared for final evaluation contained only news headlines without labels, each associated with a unique ID. Participants were expected to train their models on the standard AG News training set, perform inference on the unlabeled test set, and submit predictions in CSV format. The private leaderboard, based on a hidden subset of test samples, ensured that evaluation metrics reflected generalization ability rather than overfitting to any shared validation subset.

1.4 Our Proposed Contribution

- **Parameter-Efficient Fine-Tuning (PEFT):** We used Low-Rank Adaptation (LoRA) to fine-tune only a subset of parameters, reducing memory and compute overhead while preserving performance.
- **Roberta-Base Backbone:** A robust transformer-based encoder provided strong contextual representations, tailored for sentence-level classification tasks like AG News.
- **Selective LoRA Injection:** By restricting LoRA to specific attention modules (query, value), we stayed below 1 million trainable parameters without compromising accuracy.
- **Hyperparameter Optimization via Optuna:** We performed systematic search using HyperOpt (Optuna backend) across critical hyperparameters including learning rate, batch size, weight decay, and gradient accumulation steps guided by a Median Pruner strategy to reduce search time.
- **Final Performance:** Our model achieved ~93% evaluation accuracy during training and ~85% accuracy on the private Kaggle leaderboard, demonstrating strong generalization on real-world, unseen news data.

1.5 Challenges

A core challenge in this project was identifying the optimal set of trainable parameters while ensuring the model remained within the strict 1 million parameter limit imposed by the PEFT (LoRA) constraint. Fine-tuning transformer-based architectures such as RoBERTa required a delicate balance between expressiveness and efficiency, making the choice of r, target_modules, and lora_alpha crucial. In addition, designing an efficient Optuna-based hyperparameter search space required balancing exploration breadth with compute limitations. Tuning variables like learning rate, weight decay, batch size, and LoRA parameters (r, dropout, alpha) was critical for generalization. Hyperparameter tuning had to be performed efficiently due to limited compute, requiring careful pruning strategies and smaller subsets for search. Finally, generalizing well on the Kaggle test set despite high validation accuracy was non-trivial, emphasizing the need for robust training practices.

Methodology

The diagram below outlines our methodology, showcasing the integration of LoRA fine-tuning with RoBERTa, hyperparameter optimization, and custom data preprocessing tailored for AG News classification.

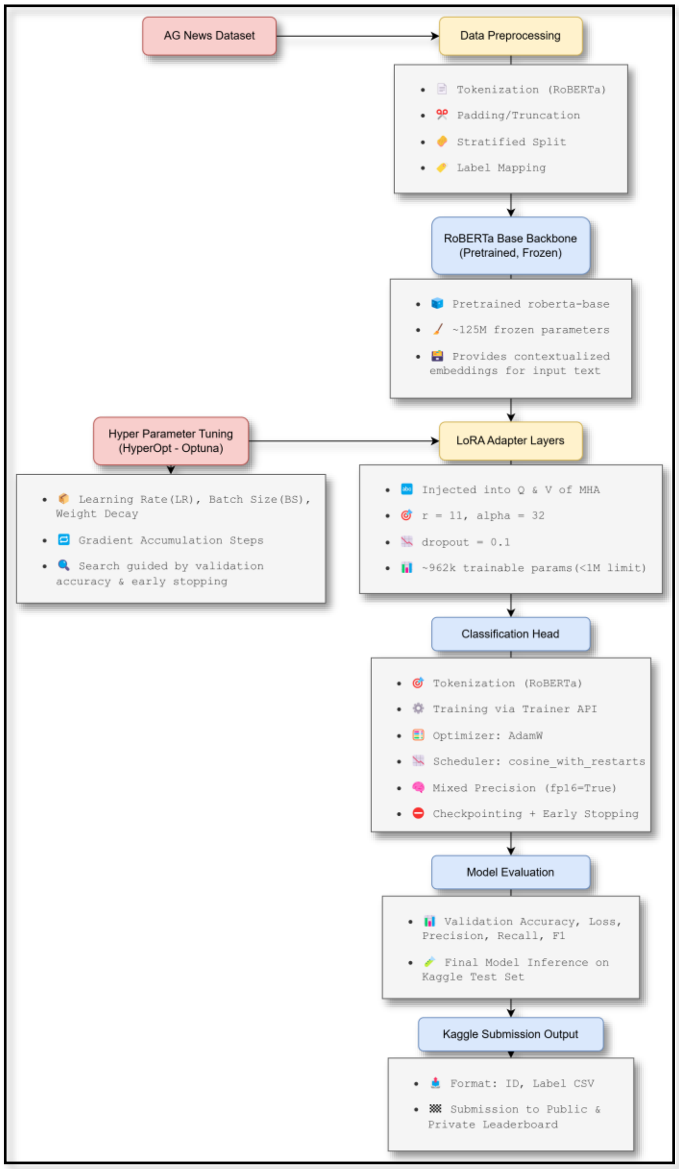
AG News Dataset
Data Preprocessing
Tokenization (RoBERTa)
Padding/Truncation
Stratified Split
Label Mapping
RoBERTa Base Backbone (Pretrained, Frozen)
Pretrained roberta-base
~125M frozen parameters
Provides contextualized embeddings for input text
Hyper Parameter Tuning (HyperOpt - Optuna)
Learning Rate(LR), Batch Size(BS), Weight Decay
Gradient Accumulation Steps
Search guided by validation accuracy & early stopping
LoRA Adapter Layers
Injected into Q & V of MHA
r = 11, alpha = 32
dropout = 0.1
~962k trainable params(<1M limit)
Classification Head
Tokenization (RoBERTa)
Training via Trainer API
Optimizer: AdamW
Scheduler: cosine_with_restarts
Mixed Precision (fp16=True)
Checkpointing + Early Stopping
Model Evaluation
Validation Accuracy, Loss, Precision, Recall, F1
Final Model Inference on Kaggle Test Set
Kaggle Submission Output
Format: ID, Label CSV
Submission to Public & Private Leaderboard

Fig. 1. System Architecture Diagram

2.1. Data Preprocessing & Augmentation

- **Tokenization:** We used the RobertaTokenizer from Hugging Face to tokenize the news headlines and bodies. Inputs were tokenized using (truncation=True) and (padding=True) to ensure uniform sequence lengths across batches.

- **Label Mapping:** The dataset's labels were automatically mapped to the labels field using Hugging Face's dataset utilities, ensuring compatibility with Trainer.
- **Train-Test Split:** We performed a stratified split using train_test_split to maintain balanced class distribution in both training and evaluation sets. For hyperparameter optimization, a smaller subset (20% train, 10% test) was used.
- **Data Collation:** DataCollatorWithPadding was employed to dynamically pad sequences to the maximum length within each batch, optimizing GPU memory and training speed.

This preprocessing pipeline ensured that the model could effectively learn from tokenized inputs while maintaining consistency and generalization across training and evaluation phases.

Pros:
- Efficient text preprocessing ensures consistent input representation for the model.
- Stratified sampling speeds up tuning while preserving class balance.

Cons
- Truncation can potentially discard valuable context in longer news articles.
- Subsampling, while practical, slightly limits generalization during tuning.

2.2. Custom LoRA-Adapted RoBERTa Architecture

To ensure parameter efficiency under the 1 million trainable parameter constraint, we adopted a Parameter-Efficient Fine-Tuning (PEFT) strategy using LoRA (Low-Rank Adaptation):

- **Backbone**: Pretrained roberta-base transformer with ~125M frozen parameters.
- **LoRA Injection:**
    - **Target Modules:** Injected into the query and value projections of attention layers.
    - **Configuration:**
        - Rank (r): 11
        - Alpha: 32
        - Dropout: 0.1
        - Bias: "none"
        - Layers transformed: All
- **Final Classification Head:** A linear layer maps the pooled RoBERTa representation to 4 output classes (one per AG News category).
- **Fully Trainable Parameter Count:** The modified model includes ~962,308 trainable parameters, satisfying the strict upper bound of 1M while retaining expressive power.

Pros
- LoRA enables efficient fine-tuning by updating a small set of low-rank matrices, reducing memory and compute costs.
- Adapting only key attention modules (e.g., query, value) preserves performance while keeping trainable parameters below 1M.

Cons
- Choosing the right LoRA configuration (e.g., rank, target modules) is non-trivial and impacts performance significantly.
- Evaluation on out-of-distribution test data (Kaggle) showed lower generalization compared to in-distribution validation results.

2.3. Training Procedure

The model was fine-tuned using Hugging Face's Trainer API with PEFT (LoRA) on the AG News dataset. Training was conducted on a Google Colab Pro+ instance with a T4 GPU. Key aspects of the training setup included:

- **Optimizer**: AdamW optimizer (adamw_torch), chosen for stability in fine-tuning transformers.
- **Learning Rate**: 3.99e-4, selected via Optuna-based hyperparameter tuning.
- **Batch Size**: 32 with gradient_accumulation_steps = 4, yielding an effective batch size of 128.
- **Scheduling**: cosine_with_restarts learning rate scheuler with a warmup ratio of 0.1.
- **Training Steps**: max_steps = 2000, with eval_steps = 400 and checkpointing every 400 steps.
- **Precision**: Mixed-precision training (fp16=True) for faster computation and lower memory usage.
- **Early Stopping**: Integrated with patience of 3 evaluations to avoid overfitting.
- **Loss Function**: Standard cross-entropy loss applied across the four AG News classes.

Hyperparameter Optimization: We integrated Optuna's hyperparameter_search() within HuggingFace's Trainer to automatically discover optimal configurations across 20 trials using a MedianPruner, focusing on maximizing evaluation accuracy.

Validation: A 5,000-image validation split was used to track accuracy and loss each epoch. The checkpoint with the highest validation accuracy was chosen for final inference on the custom test dataset.

Pros
- **AdamW** with gradient accumulation allowed efficient training on limited GPU memory without compromising convergence.
- **FP16 mixed-precision** reduced memory usage and accelerated training.

Cons
- Hyperparameter search (e.g., batch size, warmup ratio, LR scheduler) was compute-intensive and time-consuming to tune effectively.
- A fixed step budget (2000 steps) limited exploration of more extended convergence behavior.

Lessons Learned

- **Hyperparameter Optimization Matters:** Systematic tuning of learning rate, batch size, weight decay, and gradient accumulation was key in achieving optimal generalization and convergence.
- **HyperOpt Efficiency:** Instead of manual tuning, Optuna-enabled search streamlined our parameter exploration process, ensuring the best trade-offs in learning rate, batch size, and adaptation strength under the parameter budget.
- **LoRA Efficiency:** Incorporating Low-Rank Adaptation (LoRA) allowed fine-tuning of large pre-trained transformers with under 1 million trainable parameters, significantly reducing memory usage without compromising performance.
- **Data Stratification is Crucial:** Maintaining a balanced label distribution in training and validation splits prevented biased learning and ensured more stable evaluation accuracy.
- **Validation Monitoring:** Regular tracking of validation metrics helped in identifying overfitting early, allowing timely intervention using dropout and early stopping.
- **Lightweight Model Design**: Designing the parameter space to stay under 1 million required strategic control of LoRA rank (r) and targeted modules (query, value) without sacrificing classification power.

Results

3.1. Leaderboard Outcome

After fine-tuning our Roberta-based LoRA model with carefully selected hyperparameters and training strategies, we submitted the predictions on the custom test dataset provided via Kaggle. The private leaderboard reflected the following outcome:

- Accuracy: 85.325%
- Rank: 34th (Slipped from 12th to 34th after performing evaluation on 50% dataset)

3.2. Training Dynamics

- **Loss vs. Step:** Training loss showed a consistent downward trend, especially during the initial steps, with smaller drops following each learning rate adjustment. Validation loss closely mirrored this pattern, suggesting stable learning without significant overfitting. (See Fig. 2.)

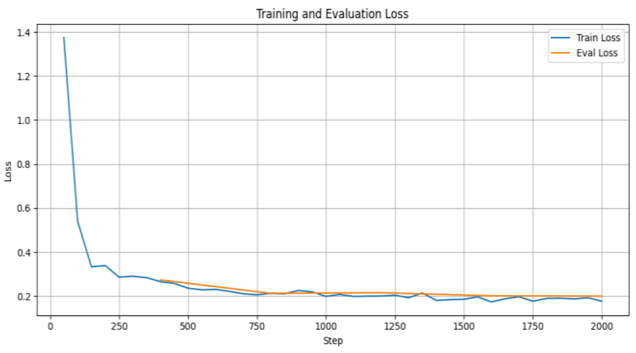

Fig. 1. Train Loss vs. Validation Loss

- **Train Accuracy & Validation Accuracy**: Accuracy improves rapidly in early epochs and stabilizes after 40 epochs. Validation accuracy closely follows training accuracy, ensuring strong generalization. (Fig. 2)

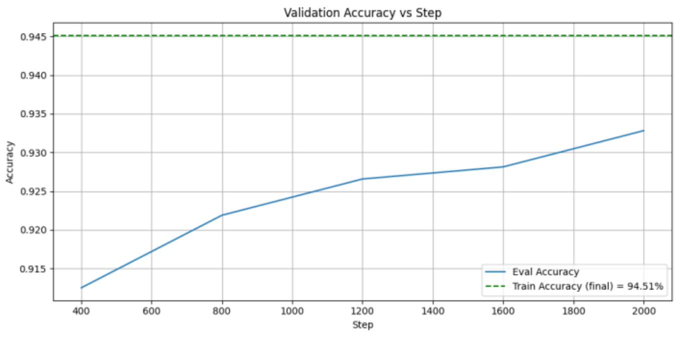

Fig. 2. Train Accuracy vs. Validation Accuracy

- **Precision, Recall, and F1 Trends:** All three metrics improved in sync with validation accuracy, confirming the model's ability to generalize well across classes. No class-specific performance drops were observed across steps. (See Fig. 3)

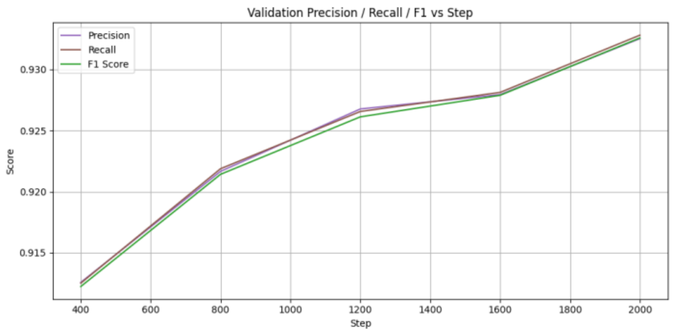

Fig. 3. Precision, Recall, F1 Trend

3.3. Comparisons

- **Full Fine-Tuning vs. LoRA:** Compared to full fine-tuning of all model parameters, our LoRA-based approach achieved comparable accuracy (~85%) while training less than 1M parameters, reducing compute cost and memory usage significantly.
- **Parameter Efficiency:** Despite using only a fraction of the trainable parameters, our fine-tuned LoRA adapter modules yielded strong performance, highlighting the

effectiveness of parameter-efficient fine-tuning for large language models in text classification tasks.

Conclusion

We developed a parameter-efficient text classification model for the AGNews dataset by leveraging the LoRA (Low-Rank Adaptation) technique over a pre-trained RoBERTa backbone. Our approach involved careful architectural decisions, aggressive hyperparameter optimization, and compliance with strict constraints keeping the trainable parameter count under 1 million.

- **LoRA Integration** enabled efficient fine-tuning by updating only low-rank weight matrices, dramatically reducing the number of trainable parameters without compromising performance.
- **Targeted Hyperparameter Search** with Optuna helped identify the optimal combination of learning rate, batch size, gradient accumulation steps, and LoRA parameters for high generalization performance.
- **Training on Public AGNews and Predicting on Private Test Data** helped demonstrate the model's real-world generalization, achieving over 85% private test accuracy.

4.1. Future Directions

- **Alternative PEFT Strategies:** Exploring other parameter-efficient fine-tuning techniques like AdaLoRA or Prefix Tuning could provide accuracy boosts under the same compute budget.
- **Data Balancing & Augmentation:** Incorporating techniques like back-translation or paraphrasing for underrepresented classes could enhance model robustness and fairness.
- **Layerwise Adaptation Control:** Applying LoRA selectively to deeper or task-specific layers may offer a more efficient trade-off between adaptation capacity and parameter count.
- **Inference Optimization:** Leveraging quantized deployment (e.g., with ONNX or TensorRT) could reduce latency without sacrificing accuracy.

We conclude that careful low-rank adaptation over strong language models, paired with targeted hyperparameter tuning, offers a competitive and efficient strategy for large-scale text classification under strict resource constraints.

Code Repository

The complete implementation, including code and training details, is available in our GitHub repository: Github Link The notebook is designed to be executed on Kaggle for ease of use and reproducibility.

References

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