# Team YAN — Report: Deep Learning Project 2: LoRA Fine tuning of RoBERTa

**Written By Team Athul, Nidhi, and Yumiko**

Kaggle team name: YAN
Repo: GitHub repository

## Overview

This project tackles the research challenge of adapting large-scale language models like BERT for efficient use in resource-constrained environments. Specifically, it explores how to maximize classification accuracy on the AGNEWS dataset using a modified RoBERTa architecture constrained to under one million trainable parameters. The core technique employed is Low-Rank Adaptation (LoRA), which introduces trainable low-rank matrices to perturb frozen model weights. This setup enables efficient fine-tuning while preserving the benefits of pre-trained representations. The project encourages systematic exploration of LoRA configurations, such as matrix rank and adaptation strength, to achieve optimal performance within the tight parameter budget of $< 1$M trainable parameters.

To identify the optimal configuration for our LoRA-augmented RoBERTa model, we conducted a comprehensive grid search across key hyperparameters. We systematically varied the LoRA rank ($r$), scaling factor ($\alpha$), learning rate, batch size, and choice of optimizer (including Adam, SGD, and Muon). For each combination, we trained the model under the constraint of fewer than one million trainable parameters and evaluated its performance on a validation split of the AGNEWS dataset. This exhaustive search allowed us to understand the interaction between low-rank adaptation settings and model generalization. We observed that lower ranks with moderate scaling values often preserved stability while enabling meaningful adaptation, and that Adam consistently outperformed other optimizers in terms of convergence and accuracy. This methodical approach ensured we selected a parameter set that struck the best balance between computational efficiency and classification performance.

The best classification accuracy was achieved using the following configuration: rank r = 4, alpha = 16, dropout = 0, bias = "lora only", optimizer = Adam, learning rate = 1e-3, batch size = 16, and epochs = 1. This setup provided strong results, and further improvements in accuracy were observed when training was extended to 8 epochs. However, due to limited compute resources and long queue times, training up to 10 epochs was not feasible within the available constraints. Despite this, the analysis demonstrates how PEFT techniques can be used to fine-tune large models effectively under resource limitations.

## Code Repository

The code repository is available at this GitHub repository

## Summary of Findings

Our project focused on fine-tuning a lightweight yet effective RoBERTa-based transformer model for topic classification on the AG News dataset using parameter-efficient transfer learning (PEFT) techniques, specifically Low-Rank Adaptation (LoRA). The primary goal was to evaluate how well LoRA can adapt large language models to downstream tasks with minimal parameter updates.

We discovered that the best test performance was achieved using a low rank, suggesting that minimal fine-tuning with LoRA was sufficient for this classification task. Interestingly, this may be attributed to similarities between the AG News test samples and synthetic AI-generated text used during pretraining, further reducing the need for extensive adaptation.

**Key findings:**

- **Rank:** The best-performing model used a rank of 4. Increasing the rank beyond this point led to a noticeable drop in performance, indicating that overly complex adaptations may lead to overfitting or reduced generalization.
- **Epoch:** Optimal performance was observed within the first few epochs of training. Extended training led to slight overfitting, as evidenced by a widening gap between training and validation accuracy.
- **Overfitting at higher ranks:** We found out that the model overfits at higher ranks, giving lower test accuracy. Models with lower ranks tended to generalize better on the AG News test set, again suggesting that minimal updates through LoRA are more effective for certain text classification tasks.
- **AI Similarity Hypothesis:** The strong performance at lower ranks might be linked to the distributional similarity between AG News content and AI-generated corpora used during pretraining. This alignment may have reduced the need for extensive fine-tuning.
- **Class Imbalance:** The class size for each class is not even. There is a high imbalance between the class sizes.

## Methodology

This section outlines the model selection, LoRA configuration, training procedures, and experimentation process used to fine-tune a parameter-efficient transformer-based classifier for the AG News dataset. Our objective was to maximize classification performance using minimal compute resources by leveraging Hugging Face's ecosystem and applying Low-Rank Adaptation (LoRA) for efficient fine-tuning of the roberta-base model.

## LoRA: Theory

### Main Idea

For a weight matrix $W \in R^{d\times k}$, LoRA modifies it like this:

$$W' = W + \Delta W$$

Instead of training $W$ directly, LoRA decomposes the **update** $\Delta W$ as:

$$\Delta W = AB$$

Where:

- $A \in R^{d\times r}$ and $B \in R^{r\times k}$
- $r$ is a small **rank** (usually 1–64), so $AB$ is a low-rank approximation of $\Delta W$
- $W$ is frozen, $A$ and $B$ are the **only trainable parameters**

Often this is done with a scaling factor $\alpha$ (LoRA alpha):

$$\Delta W = \frac{\alpha}{r} AB$$

### 1. Model Selection and LoRA Integration

We began by selecting roberta-base, a pre-trained transformer model with strong generalization capabilities on natural language tasks. Instead of full fine-tuning, we adopted LoRA to update only a small set of trainable parameters within attention layers. This significantly reduced memory usage and training time.

LoRA Configuration: The best setup used rank r = 4, alpha = 16, dropout = 0, and bias = ”lora only”. This configuration allowed for efficient adaptation with minimal overhead. LoRA modules were injected into the attention layers using Hugging Face's peft library, and the rest of the model was kept frozen to avoid overfitting and reduce compute requirements.

### 2. Dataset Preprocessing and Tokenization

The AG News dataset was loaded via the Hugging Face datasets library. Preprocessing included:

Tokenization using roberta-base tokenizer with truncation and padding applied dynamically during batching.

Label extraction and mapping from string to integer IDs using the ClassLabel feature.

Data collation handled via DataCollatorWithPadding to ensure consistent input lengths during training.

This setup ensured robust preprocessing while maintaining compatibility with the model's input expectations.

## 3. Training Configuration and Strategy

Training was conducted using Hugging Face's Trainer API, which simplified training and evaluation workflows.

Hyperparameters (Best Performing Run):

Optimizer: Adam

Learning Rate: 1e-3

Batch Size: 16

Epochs: 1 (due to resource constraints)

Weight Decay: 0

Scheduler: Linear with warmup disabled Despite the short training cycle, this configuration yielded strong results. Increasing to 8 epochs showed better performance, but high queueing times and limited GPU availability prevented further scaling to 10 epochs.

## 4. Evaluation Metrics and Checkpointing

Accuracy was used as the primary evaluation metric, computed using Hugging Face's evaluate module. Training checkpoints and logs were automatically saved, allowing model recovery and future experimentation. LoRA adapters were saved separately from the full model to preserve modularity and reduce storage overhead.

## 5. Experimental Trade-offs and Resource Management

While longer training epochs improved accuracy, constraints such as timeouts and limited free-tier GPU access made sustained training impractical. Our design prioritized reproducibility, efficient memory usage, and fast convergence, all of which were achieved through LoRA-based fine-tuning.

## 6. Inference and Post-Training Analysis

Post-training, predictions were decoded and mapped back to class labels to assess qualitative performance. Model outputs were consistent with training labels, and evaluation metrics confirmed good generalization on the validation set despite minimal compute usage.

## Experiments

### Model Search

We began with the standard roberta-base model as the backbone due to its robust performance on NLP benchmarks and its compatibility with parameter-efficient fine-tuning methods. Our baseline experiment involved applying LoRA to selected attention layers while freezing the remaining weights. We conducted several iterations with varying LoRA parameters (r, alpha, dropout, bias) to determine the optimal configuration. After extensive tuning, the setup with r = 4, alpha = 16, dropout = 0, and bias = ”lora only” yielded the best trade-off between performance and efficiency.

In addition to LoRA-specific tuning, we explored the impact of increasing the number of training epochs. While our initial experiments were limited to a single epoch due to resource constraints, training for up to 8 epochs led to improved accuracy. Unfortunately, extending to 10 epochs was not feasible due to high queue times and limited compute access.

### Data Preparation

The AG News dataset was loaded using the Hugging Face datasets library, which provided clean and consistent splits for training and validation. Text samples were tokenized using the roberta-base tokenizer, with truncation and padding handled dynamically via DataCollatorWithPadding. This allowed us to batch sequences of varying lengths without manual padding logic. Class labels were automatically extracted and mapped using the dataset's built-in metadata, ensuring correct alignment during training and inference.

No additional data augmentation was applied, as the dataset consists of textual data. However, we ensured consistent preprocessing across all experiments to isolate the effects of LoRA configuration and training dynamics.

### Optimizers and Schedulers

For optimization, we selected the Adam optimizer due to its strong performance with transformer-based models and its stability in low-resource training settings. The best results were obtained using a learning rate of 1e-3 without any weight decay. Although schedulers like ReduceLROnPlateau and linear warmup were considered, we ultimately disabled schedulers to simplify training and reduce tuning complexity. This decision proved effective for our short training cycles.

The Hugging Face Trainer framework managed gradient accumulation, logging, and checkpointing, streamlining experimentation. In each run, accuracy was used as the primary metric, evaluated at regular intervals to track model improvement over time.

## Results

The resulting model is a low-rank adaptation (LoRA)-tuned vision transformer model trained on top of a pre-trained backbone. The model incorporates LoRA modules, which enable parameter-efficient fine-tuning by injecting trainable low-rank matrices into specific layers, reducing the number of updated parameters during training.

The best-performing configuration was found with the following hyperparameters: a LoRA rank of 4, alpha set to 16, dropout rate of 0, and bias mode set to `lora_only`. The model was optimized using Adam with a learning rate of $1 \times 10^{-3}$, a batch size of 16, and trained for 1 epoch.

- The model achieved its best test accuracy under the configuration: $r = 4$, $\alpha = 16$, dropout = 0.
- Only LoRA-specific parameters were updated during fine-tuning (`lora_only` bias mode).
- Training was done using the Adam optimizer, batch size of 16, and a learning rate of $1 \times 10^{-3}$.
- The peak accuracy was observed during the 8-epoch run, although due to computational constraints and queue times, training could not be extended to the full 10 epochs.

Despite the limitation in training time, the model showed promising performance gains with increased epochs, suggesting that further fine-tuning could lead to even better accuracy. This demonstrates the effectiveness of LoRA for efficient transfer learning, especially under constrained resource environments.

### Results Summary

The performance of various model configurations is summarized below:

## Citations

1. Kaiming He et al., ”Deep Residual Learning for Image Recognition”, CVPR 2016.
2. Jie Hu et al., ”Squeeze-and-Excitation Networks”, CVPR 2018.
3. Alex Krizhevsky, ”Learning Multiple Layers of Features from Tiny Images”, Technical Report, 2009.
4. Diederik Kingma and Jimmy Ba, ”Adam: A Method for Stochastic Optimization”, ICLR 2015.
5. H. Robbins and S. Monro, ”A Stochastic Approximation Method”, Annals of Mathematical Statistics, 1951.
6. L. N. Smith, ”Cyclical Learning Rates for Training Neural Networks”, 2017 (for LR scheduling inspiration).