Deep Learning Project 2
Spring 2025

Agustin Leon (al8937), Akhil Manoj (am14580), Anup Raj Niroula (arn8147)

[Code repository link]

Introduction

In this project, we explored the potential of fine-tuning text classification models with Low-Rank Adaptation (LoRa), for a Kaggle competition of the Deep Learning course of NYU Tandon. The challenge was to fine-tune a RoBERTa model on the AG-News dataset using LoRa, restricted to having no more than 1 million parameters. Our implementation got 92.83% validation accuracy, and 86.95% test accuracy on an out-of-distribution (OOD) held-out test set.

Our research process consisted mostly of two stages: first, we initiated the work conducting a series of experiments of different training jobs tuning mostly hyperparameters related to optimization (learning rate, batch size, weight decay, etc.), and also tuning parameters specific to LoRa (rank, alpha, biases, etc.). This allowed us to get to something close to 85% test accuracy, but then we couldn't improve further. The second stage was exploring more advanced techniques to push our model a bit further. We tried data filtering, some basic synonym-based data augmentation, and knowledge distillation. While these techniques helped a little, they did not produce a significant difference. The breakthrough came from trying a more agressive data augmentation technique based on domain adaptation by fine-tuning a GPT-2 model and generating synthetic samples with it. Our test accuracy increased significantly when we trained our RoBERTa model on this augmented dataset.

Methodology

Model Architecture:

We used the "RoBERTa-Base" (Liu and et al. 2019) as the starting point for the model. To satisfy the constraint of having fewer than 1 million training parameters, we have implemented low-rank adaptation (LoRA). We utilized the PEFT library(Mangrulkar, Gugger, and et al. 2022) to implement this technique.

LoRA Configuration: Our approach to ensuring the RoBERTa model is under the 1-million trainable parameter limit involved analyzing the LoRA configuration and identifying ways to cut down the complexity while maintaining performance. We decided to test different combinations of LoraConfig configuration to observe the number of trainable parameters in the model and the performance (accuracy) obtained when training the model. We discovered the LoraConfig parameters that directly affected the number of parameters and performance of the model were: `r` (rank of the matrix); `lora_alpha` (Strength of update α); `lora_dropout` (dropout of LoRA); `target_modules` (target modules); `bias` (bias); `task_type` (Task Type).

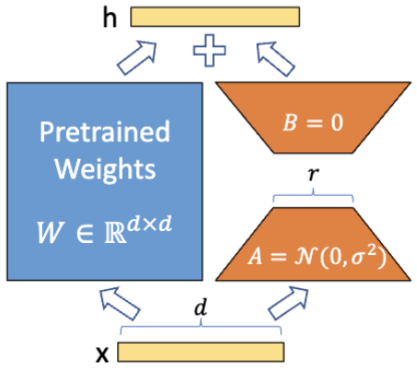


Figure 1: *Block Diagram of LoRA. Low-rank matrices A and B are introduced and trained, while the pretrained weight matrix W remains frozen. This allows for efficient fine-tuning with a reduced number of trainable parameters.*

Rank of the Matrix: This defines the rank of low-rank matrices. Increasing the rank significantly increased the number of trainable parameters. Through trying out different values, we found out that the optimal value for rank was 4, which satisfied our target of less than 1M parameters.

Alpha α: This parameter is used to keep the magnitude of LoRA-updated weights comparable to the pre-trained weights for stable learning. We found that alpha of 64 gave us a balance of regularization and have effective learning. This gave a scaling factor of $\frac{\alpha}{r} = 16$.

Dropout: As we are training less than 1 million parameters, we added dropout to prevent overfitting. We chose a moderate dropout rate of 0.1.

Target Modules: This parameter specifies which modules in the model should be modified (adapted) with LoRA adapters during fine-tuning. We wanted the model to adapt to new tasks with minimal updates, so we specified the target modules to be "query", "key", and "value". This ensured we

kept low trainable parameters, while keeping efficient training.

Bias: There exists a possibility of loss of expressiveness and flexibility of the model due to the low-rank of the weight updates. To compensate for this, we specified that all the biases of the target modules should be trained. This helped us gain more flexibility in the model for a very small increase in parameter count, which we considered a fair trade-off.

Task Type: Since our model had the task of classifying the news dataset, we chose the task type to be sequence classifier ("SEQ_CLS"), so that the PEFT library understands that we are doing a task where input is a sequence and output is a class label.

From the configuration specified above, we got a model with trainable parameters of 916,996 out of 125,463,560 total parameters. So, we had 0.7309% trainable parameters. As we see in the later section, this configuration resulted in a highly accurate model which achieved a test accuracy of 92.83% in our dataset.

`LoraConfig params`	**Values**
r	4
lora_alpha	64
bias	all
target_modules	[query, key, value]
task_type	SEQ_CLS

Table 1: *LoRA Configuration for the model*

Training Arguments: With the Training Arguments parameters, we focused on the model with the best performance gain and reasonable training time. We explored different hyperparameter settings, including batch size, learning rate, and weight decay, and warm-up ratio. We tested training batch sizes of 16, 32, and 64, finding that 64 provided the most stable and efficient training process. We multiplied the evaluation batch size by 4 times the training batch size. Learning rates were also tuned, with 2e-5 proving to be the most effective for convergence. Weight decay of 0.01 was utilized to prevent overfitting, ensuring that the model focused on meaningful features rather than memorizing the training data. We also added a warm-up ratio of 0.1 so that there is the gradual increase of learning rate from 0 to our value of 2e-5. We did this to prevent sudden and large updates early in the training which could potentially make the learning unstable. The hyperparameters used are described in Table 2:

Training Arguments	**Values**
Batch Size	64
Learning Rate	2e-5
Weight Decay	0.01
Warmup ratio	0.1

Table 2: *Training Arguments for the model training*

Data Augmentation:

As mentioned in the introduction, the major breakthrough in our model performance was due to the introduction of domain adaptation data augmentation, via the use of fine-tuned LLM models for augmented sample generation. We reviewed some articles and papers such as (Li and et al. 2024), (Murray 2024), (Dingliwal and et al. 2022), and (Bismi 2023).

Initial data augmentation with synonyms and special characters handling: Upon inspecting samples from the training data, we observed that many examples contained encoded HTML and escape characters, which could hinder the model's ability to generalize effectively. To address this, we parsed and removed these unwanted characters during both training and inference on the hidden dataset. Additionally, we experimented with truncating the text, hypothesizing that shorter, cleaner inputs might improve generalization. We also applied synonym-based augmentation using the WordNet lexical database, generating semantically similar sentences to enhance model robustness. These preprocessing and augmentation steps yielded a modest improvement in accuracy, indicating the need to explore more aggressive techniques.

Original Text	**Augmented Text (WordNet)**
Fed to Raise US Rates to Underpin Steady Growth Another hike in US interest rates is seen as a certainty when Federal Reserve policy-makers meet on Wednesday – a step on the march toward a level of rates necessary to underpin long-term expansion.	Fed to Raise US Rates to Underpin Steady Growth Another hike in US pastime rates make up seen as a certainty when Federal Reserve policy - makers fill on Wednesday - - a step on the mar toward a level of rate necessary to bear out long - condition expansion.
Shia militias clash in Lebanon Two rival Shia militia groups clash in a town in southern Lebanon and army forces are called in to calm the situation.	Shiah militia clash in Lebanon II rival Shia militia groups clash in a town in southern Lebanon and regular army force out are called in to calm the situation.

Table 3: *Original samples from the AGNews dataset and Augmented samples from SynonymAug with WordNet dictionary*

Final data augmentation based on domain adaptation with LLM models: By exploring the unlabeled data, we noticed that the vocabulary was quite different to the AG-News dataset. We also noticed that there were some particularly difficult-to-classify samples, that blended the four classes of AG-News (for example, a business news about the sports industry, or a global news about a the Mad Cow disease). We realized about the need to augment the data in order to allow our model to generalize better to this OOD test

dataset. While we could have blended several augmentation techniques, and potentially arrived to a similar (or better) performance, the idea of using an LLM to generate sample sounded interesting. We did some research about it and decided to give it a try. We will next describe our implementation.

1. The first step was to grab a GPT-2 model from Hugging Face and lightly fine-tune it on the unlabeled samples for only two epochs. The point of this is to 'infuse' the model in the vocabulary and contextual representations that are present on the OOD test set. However, we don't want to fine-tune it for too long, since generalization capacity of the GPT-2 model needs to be preserved. Two epochs were enough to achieve the desired result.

2. The second step is to generate samples with the fine-tuned GPT-2 model. We were very careful here.

   (a) First, we hand-wrote between 15-20 different prompts for each of the AG-News classes. A few examples: "Cover a story from Middle East or Western Asia:", "Summarize a finding in biology:". We also included more nuanced prompts that tried to generate difficult samples: "Write about a business article on the sports industry:", "Write a tech or science article with business-related words". The label for each sample will of course be assigned depending on the class that the prompt belonged to.

   (b) Second, we wanted to emulate the sample length distribution of the training data, as opposed to generating samples with a fixed length. For that we basically plotted an histogram and eyeballed its distribution. Then we randomized the sample length from a skewed normal distribution with min=1 and max=128 (we didn't want samples longer than 128 words mostly for efficiency).

   (c) Third, we also used some parameters that are provided by Hugging Face for its generator pipeline. The most important one is the repetition_penalty parameter, that punishes the LLM for repeating itself too much from sample to sample. We tried a few configurations, and finally set that parameter to 1.3.

3. Having configured the sample generator, we then proceeded to generate 12,000 augmented samples, 3,000 per class. Then we appended the augmented samples to the AG-News dataset and shuffled it. Our final dataset went from 120,000 to 132,000 samples. We are including below some examples of the generated samples.

```
🌎 Label 0: World

Sample 1:
Iran #39;s top diplomat, Saifollah Khomeini has
been sentenced to four years in prison for spying
on American officials and diplomats. The former
foreign minister was accused of secretly obtaining
confidential US intelligence about Tehran's
activities after they discussed  quot;nuclear

Sample 2:
The New York Times, the Washington Post and Foreign
Affairs all said that they had met with
quot;Furter to discuss U.S.-led economic sanctions
against Saudi Arabia on Tuesday in the wake of
ongoing oil prices drop after their leaders gave an
upbeat speech.

⚽ Label 1: Sports

Sample 1:
A video is shown of former gymnast Elin Kornfeld
performing. At the end, with her two men holding
hands in front to give their support and cheer

Sample 2:
On one side is basketball and there will be more.
In baseball, there was the All-Star Game on Monday
night because of a team injury but also the San
Diego Padres that won the
```

Figure 2: *Generated samples for 'World' and 'Sports'*

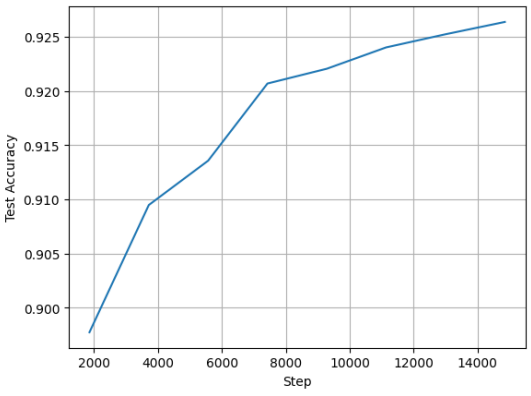
```
💼 Label 2: Business

Sample 1:
Amazon.com shares closed higher Tuesday afternoon
after Oracle Corp., the world #39;s largest
corporation, confirmed it will halt research into
its software and cloud

Sample 2:
Apple.com reported yesterday that new CEO Bill
Gates will join Intel in the board of directors,
creating an impressive resume for any company
executive who wishes to grow technology within
their organization's ranks -- despite prior
acquisitions and reorganizations underway across
other tech

🧪 Label 3: Sci/Tech

Sample 1:
Could have some sort of machine, or anything at
all. At least it seems like most AI researchers
that would say that something may be coming sooner
than you think is possible (I

Sample 2:
Astronauts found water on Mars that was previously
thought to exist in an ancient fossil lake. Since
taking part as astronauts, they have successfully
re-examined it extensively from one extreme end of
the
```

Figure 3: *Generated samples for 'Business' and 'Sci/Tech'*

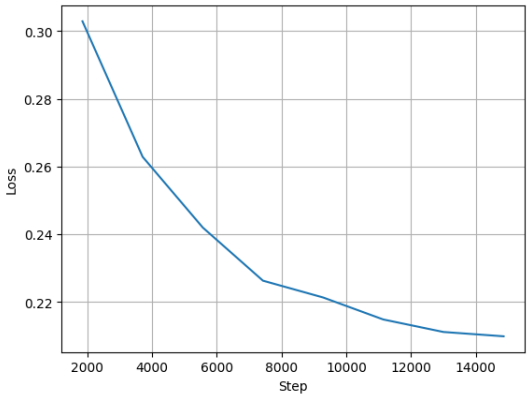
Model Training:

Training Environment: The model training was done primarily in HPC to allow faster training speed.

Optimizer: In our implementations, we started with using the AdamW optimizer(Loshchilov and Hutter 2019), which



(a) *Accuracy vs Step*



(b) *Loss vs Step*

Figure 4: *Accuracy and Loss Curve*

is a battle-tested optimizers used for training transformer models. Later on, we opted to use the AdamW Fused optimizer, to improve the training time and memory footprint for the same performance as AdamW optimizer.

Logging: We used tensorboard (Abadi, Agarwal, and et al. 2015) to log every experiment and run we did for this project. This enabled us to easily log the training data without any extra code.

Checkpoint: We used the `TrainingArgument`'s default checkpoint implementation to save the model after every epoch.

Training job description: In all training jobs, it was observed that the testing accuracies increased rapidly until they reached 90% and slowed significantly. In the best training run, the training and test accuracy reached about 91% by the first 3 epochs. After that, the accuracy increased extremely slow, around 1% for next 7 epochs. After this, the accuracy gain was negligible, so the training was stopped after 11 epochs. The observed trends are captured in the graphs above.

Conclusions

Throughout this project, we realized about the impressive capacity of LoRa fine-tuning for keeping large models performance in much lighter trimmed-down packages, with a very light cost in accuracy. With our implementation we were able to bring down the parameter count from a RoBERTa model down to 916,996 trainable parameters, achieving 93.83% validation accuracy and 86.95% held-out test accuracy.

The main strengths of our approach were two. First, we reached a solid configuration of hyperparameters for both the optimizer and the LoRA-specific settings through experimentation. Second, we added a bold data augmentation strategy that helped our model adapt better to the style and vocabulary of the unlabeled data. This was done by generating synthetic samples using a lightly fine-tuned GPT-2 model.

To improve further, we could try experimenting more with the sample generation process. For example, we could incorporate a larger amount of synthetic samples, with more diverse and specific prompts, to help the model learn from a wider range of cases and generalize even better.

Tools and Libraries

- Our code implementation was initially based on the provided starter notebook.
- PyTorch documentation.
- Hugging Face documentation. Particularly documentation about the PeftModel functionality.
- Tensorboard documentation.
- **ChatGPT 4.0**: we used the ChatGPT 4.0 LLM model for learning about LoRa and the math behind it. We also requested code for implementing the generator pipelines from Hugging Face. We also used ChatGPT for some of the plots to better understand the data distribution.

References

Abadi, M.; Agarwal, A.; and et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

Bismi, I. 2023. Augmenting Text using Large Language Models: GPT-2, GPT-3, BERT. Accessed: 2025-04-21.

Dingliwal, S.; and et al., A. S. 2022. Prompt Tuning GPT-2 language model for parameter-efficient domain adaptation of ASR systems. Accepted at Interspeech 2022, arXiv:2112.08718.

Li, Y.; and et al. 2024. Data generation using large language models for text classification: An empirical case study. Accessed: 2025-04-21, arXiv:2407.12813.

Liu, Y.; and et al. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, abs/1907.11692.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. arXiv:1711.05101.

Mangrulkar, S.; Gugger, S.; and et al. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. https://github.com/huggingface/peft.

Murray, C. 2024. Generating synthetic data with LLMs for fine-tuning. Accessed: 2025-04-21.