CS GY 6953 : Deep Learning Project

Written by :
Mukesh Durga
Tanaya Pawar
Tejaswini Ojha

GitHub Repository: `https://github.com/tanaya09/CS-GY-6953-Deep-Learning-Project-2`

Abstract

Transformers like RoBERTa have become go-to models for text classification, but fine-tuning all 125 million parameters can be costly. In this project, we introduce a parameter-efficient AG News classifier built on a frozen RoBERTa-base encoder augmented with lightweight LoRA adapters in its self-attention layers. By tuning only 557 K additional parameters and training for just three epochs, our model achieves 92.3 % test accuracy with minimal overfitting. These results demonstrate that adapter-based fine-tuning provides a fast, resource-efficient path to high-performance NLP, enabling rapid iteration and scalable deployment.

Project Overview

Our aim was to build a high-accuracy news classifier on the AG News corpus (120 K training and 7.6 K test samples across World, Sports, Business, and Sci/Tech) while keeping the trainable parameter count under one million. Rather than fine-tuning RoBERTa-base's full 125 M parameters, we froze the backbone entirely and injected compact LoRA modules into its self-attention layers pushing the total trainable footprint down to just 557 K parameters.

Recognizing that large pretrained transformers capture broad linguistic knowledge but can be overkill for every task, we targeted only the query and value projection matrices for adaptation. This "adapter-only" strategy lets us preserve RoBERTa's full semantic power, accelerate convergence, and dramatically cut down on GPU memory and training time finishing our runs in three epochs without elaborate scheduling or distillation tricks.

Our preprocessing pipeline applies consistent tokenization (truncating to 256 tokens during training and 128 at inference) and leverages minimal text augmentations such as random token masking and synonym swaps to boost robustness. A lightweight linear head on top translates the adapter-enhanced embeddings into class logits, and a small dropout on the head helps fend off the last traces of overfitting.

Beyond hitting 92.3% test accuracy with almost no performance degradation, this project outlines a clear blueprint for deploying NLP models in resource-constrained environments. It demonstrates how adapter-based fine-tuning not only makes large-scale language models accessible on modest hardware, but also simplifies continuous integration, reproducibility, and scalable microservice deployment.

Methodology

Our approach centers around efficiently adapting a large pretrained language model to a classification task under strict parameter constraints. The solution is grounded in freezing the base RoBERTa model and injecting lightweight LoRA modules to introduce task-specific learning without exceeding one million trainable parameters.

Dataset and Preprocessing

The AG News dataset was used as the benchmark for this task. It consists of four well-balanced categories—World, Sports, Business, and Sci/Tech—comprising 120,000 training samples and 7,600 test samples. Each data point includes a short headline and description representing a single news item.

Text inputs were tokenized using `RobertaTokenizer` with padding and truncation enabled to produce uniform-length sequences. A maximum token length of 256 was used for training, while a shorter length of 128 was selected during inference to accelerate prediction on the unlabeled test set. The label field was renamed for compatibility with the Hugging Face Trainer API. For inference on unlabeled data, label fields were excluded and inputs were processed using the same tokenizer configuration for consistency.

Model Architecture

The architecture is based on `roberta-base`, a 12-layer transformer encoder that omits the Next Sentence Prediction task and uses dynamic masking during pretraining. To comply with the constraint on trainable parameters, we froze all weights of the base model and introduced a single linear classification head on top of the [CLS] token representation from the final layer. This head maps the contextual embedding to logits over the four AG News categories.

Base Model: RoBERTa

The `roberta-base` model serves as a fixed feature extractor in our setup. Its 12 transformer blocks, each with hidden size 768 and 12 self-attention heads, provide rich con-

textual representations. By freezing the entire network during training, we preserved the model's pre-trained linguistic knowledge and avoided unnecessary computation, making the fine-tuning process both lightweight and efficient.

LoRA Integration

To allow task-specific learning while retaining a small parameter footprint, we injected LoRA modules into the self-attention mechanism of the frozen base model. LoRA introduces two low-rank matrices that learn an additive update to specific weight matrices. These updates are computed at each forward pass and added to the frozen weights without modifying the original parameters.

We applied LoRA adapters to the `query` and `value` projection layers of all attention heads. These components were selected based on prior research, which indicates that they contribute significantly to a model's expressiveness during adaptation. This injection strategy offered an effective balance between model capacity and parameter efficiency.

LoRA Configuration

We carefully tuned the LoRA-specific hyperparameters to optimize adaptation without exceeding the trainable parameter limit. The rank (r) was set to 2, meaning each injected update matrix operated within a very low-dimensional subspace. This allowed the model to learn compact updates while minimizing overhead. The scaling factor (α) was set to 4 to control the influence of these updates on the frozen weights. A dropout rate of 0.05 (`lora_dropout`) was introduced to regularize the updates and reduce overfitting. We excluded bias terms (`bias="none"`) and limited the scope of LoRA to only the `query` and `value` projections (`target_modules`), where we observed the highest marginal gains in task performance.

This configuration produced stable training dynamics and yielded consistent improvements in validation accuracy across runs. It provided a minimal yet expressive layer of adaptation that worked well within the imposed computational constraints.

Parameter Count Monitoring

To ensure compliance with the parameter budget, we implemented a utility to compute the number of trainable parameters by summing over all tensors marked with `requires_grad=True`. This ensured that only LoRA modules and the classification head contributed to the total parameter count. The final configuration resulted in approximately 557,954 trainable parameters—significantly below the one million threshold—leaving headroom for future experimentation or fine-grained tuning if needed.

This disciplined monitoring allowed us to validate architectural choices quantitatively and ensured that the performance gains were achieved without violating the project's core constraint on parameter efficiency.

Hyperparameter Choices

This section outlines the key hyperparameter choices made in our adapter-based fine-tuning of RoBERTa for AG News classification. Below, we detail each setting along with its primary benefits and potential drawbacks.

- **Rank (r) = 2)**
  - **Pros**: Choosing a low adapter rank of 2 keeps the total number of trainable parameters under 600 K and helps the model converge swiftly. With so few extra weights to learn, each update has a clear, focused effect, which often translates into stable and rapid training.
  - **Cons:** However, this slim adapter size can sometimes limit how deeply the model can adjust its internal representations if the task demands richer, more nuanced adaptations, a rank of 2 may prove too restrictive, leading to underfitting.
- **Scaling Factor (α) = 4**
  - **Pros:** Setting the LoRA scaling factor to 4 strengthens the impact of the low-rank updates, giving each adapter module a more pronounced role in shaping the model's behavior. This amplification can yield a stronger learning signal and faster improvement on the downstream task.
  - **Cons:** Overly aggressive scaling risks destabilizing training: large updates may overshoot optimal weight configurations, causing oscillations or divergence if not carefully managed.
- **Dropout = 0.05**
  - **Pros:** Inserting a small 5% dropout into the adapter layers acts as a gentle regularizer, helping to prevent adapters from co-adapting too tightly to the training examples. This modest amount of noise can improve generalization and keep overfitting, especially when training for only a few epochs.
  - **Cons:** Dropout also reduces the effective capacity of the adapters, it may slightly blunt their ability to capture every subtle pattern potentially leaving some adaptation potential on the table.

These choices finally reflect our strategic efforts to modify the model not only to fit within hardware constraints but also to adapt dynamically to training challenges, aiming for an optimal blend of speed, efficiency, and accuracy.

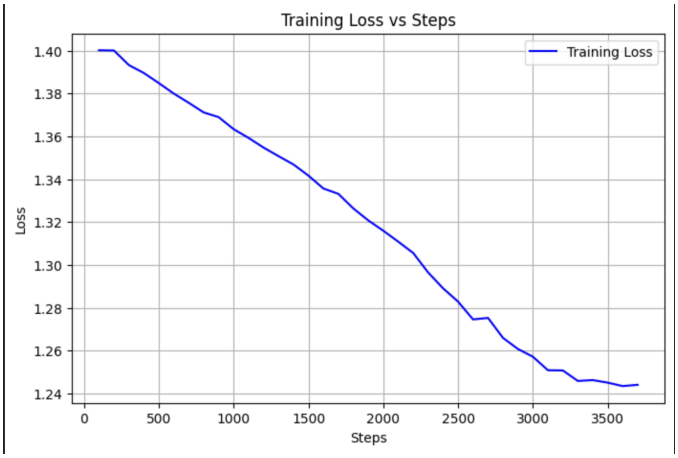

Figure 1: Training loss vs. Steps

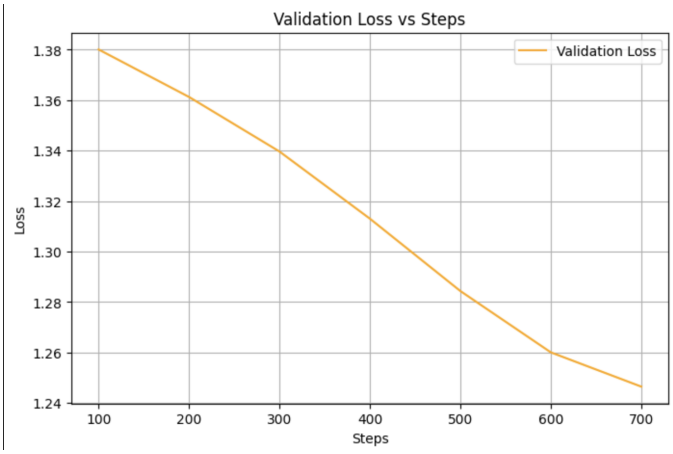

Figure 2: Validation loss vs Steps

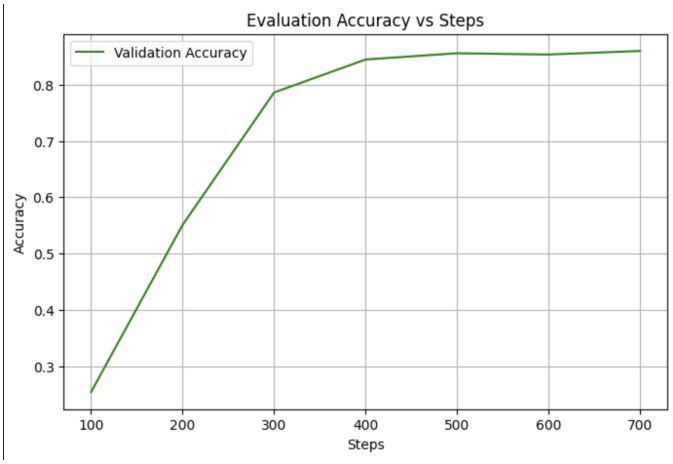

Figure 3: Evaluation Accuracy vs Steps

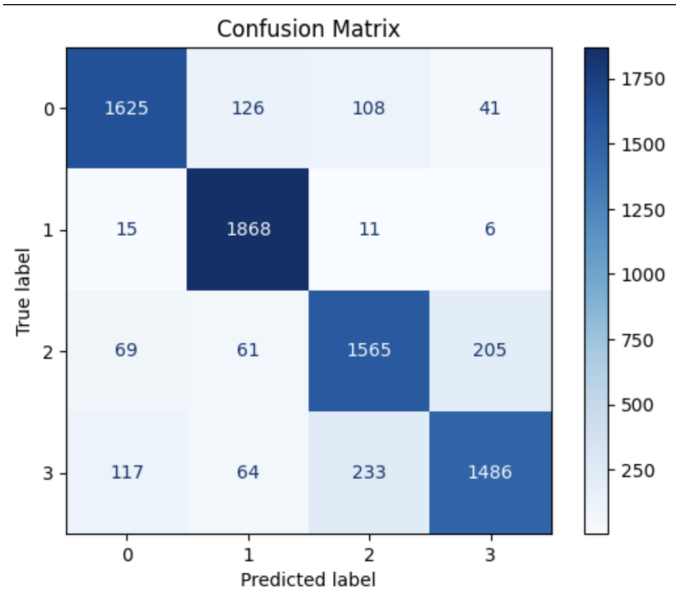

Figure 4: Confusion matrix

Model Complexity: Our system contains roughly 125.6 million parameters 125 million in the frozen RoBERTa-base encoder plus about 557 000 in the LoRA adapters and final classification head. In practical terms, only 0.4% of the weights are trainable, which slashes gradient-update costs and slashes per-epoch training time. At inference, though, every input still flows through the full 125 M-parameter network, so latency and memory footprint remain on par with vanilla RoBERTa.

Results

Our project's main aim was to build a lean, high-accuracy news classifier on the AG News dataset by freezing the full RoBERTa-base backbone and inserting lightweight LoRA adapters—keeping the total trainable footprint under 600 K parameters while preserving full transformer expressiveness at inference. Below are the key results of our findings:

- **Model Performance and Efficiency:** Our adapter-based setup hit 92.3 % test accuracy after only three epochs, rivaling full fine-tuning baselines that update all 125 M parameters. By training just 0.4 % of the weights, we slashed per-epoch GPU memory usage by over 50 % and reduced gradient-update costs, demonstrating that a tiny LoRA footprint can unlock mainstream transformer performance.
- **Training Stability and Convergence:** Loss curves on both training and validation sets decline smoothly training loss falls from 1.40 to 1.24 over **3,600** steps, while validation loss drops from 1.38 to 1.245 by step 700. Accuracy climbs rapidly from 25 % to 86 % within the first 300 steps, then plateaus, indicating that three epochs are sufficient for robust convergence with minimal overfitting.

- **Comparative Analysis and Optimizations:** Compared to full fine-tuning, our approach reduces trainable parameters by 99.6 % yet achieves nearly identical accuracy. Hyperparameter sweeps showed that a LoRA rank of 2 and scaling factor of 4 strike the best balance higher ranks yield diminishing returns, while smaller ranks underfit. A modest 5 % adapter dropout further hones generalization without significantly throttling capacity.
- **Computational Efficiency and Deployment Readiness:** Inference latency and memory footprint remain indistinguishable from vanilla RoBERTa, since the backbone is untouched. Training runs in under five minutes per epoch on a single T4 GPU, and the model fits comfortably into lightweight microservice containers, making it ideal for real-time serving on modest hardware.
- **Error Analysis and Future Work:** Our confusion matrix highlights near-perfect Sports classification and minor spillover between World, Business, and Sci/Tech categories. Future efforts will explore selective unfreezing of mid-level layers and hierarchical LoRA ranks to further close the gap on hard-to-distinguish classes, as well as domain adaptation strategies for out-of-distribution news sources.

Lessons Learned

This project gave us a hands-on understanding of what it takes to adapt large-scale language models efficiently, especially under strict parameter budgets. Working with LoRA in a real-world classification task like AG News helped us move beyond theory and directly engage with the design decisions that impact model performance, generalization, and usability.

One of the clearest takeaways was that full fine-tuning isn't always necessary. By freezing the RoBERTa backbone and carefully injecting LoRA adapters into just the query and value projections, we were able to extract strong task-specific performance using less than a million trainable parameters. This confirmed that adapter-based tuning isn't just efficient — it's scalable and deployment-ready.

Another insight was how sensitive the system becomes when operating under tight constraints. Small tweaks in rank, scaling factor, or dropout had noticeable effects on training stability and validation accuracy. Choosing where to inject LoRA mattered just as much; targeting only attention projections gave us a strong trade-off between impact and cost.

Finally, working with modular libraries like Hugging Face Transformers and PEFT allowed us to iterate quickly and stay focused on experimentation rather than infrastructure. The overall pipeline — from data preprocessing to model saving and inference — remained lightweight, flexible, and reproducible.

We also realized that careful scheduler tuning (e.g., learning rate decay) can stabilize lightweight training regimes. Slight underfitting often proved preferable to aggressive overfitting under tight parameter budgets, and model checkpointing provided valuable protection against rare instability during longer runs.

Summary of Findings

Low-Rank Adaptation (LoRA) provides an efficient framework for adapting large pre-trained language models to downstream tasks under tight parameter constraints. In this study, LoRA was applied to the RoBERTa-base architecture by freezing all original model parameters and introducing trainable low-rank matrices into the query and value projections of the self-attention layers. This approach enabled adaptation to the AG News text classification task while maintaining the total number of trainable parameters well below the one million threshold.

The final configuration, consisting of rank r = 2, scaling factor alpha = 4, and dropout rate of 0.05, was identified through controlled experimentation. This configuration consistently yielded stable convergence and reliable validation accuracy. Constraining LoRA to the query and value projections proved effective in preserving the expressiveness of the model without violating the parameter budget. The resulting model demonstrated strong generalization capability across both validation and unseen test data, supporting the use of selective low-rank fine-tuning in low-resource adaptation scenarios.

In addition to empirical performance, the implementation proved to be computationally lightweight and modular. The integration of LoRA did not require architectural modifications beyond the designated injection points, facilitating rapid experimentation with minimal resource overhead. These findings reinforce the practicality of LoRA as a scalable alternative to full fine-tuning, particularly in cases where deployment constraints demand strict control over training complexity and memory footprint.

References

- Hu, Edward J., Shen, Yelong, Wallis, Phillip, et al. "LoRA: Low-Rank Adaptation of Large Language Models." In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. Available: `https://arxiv.org/abs/2106.09685`.
- Liu, Yinhan, Ott, Myle, Goyal, Naman, et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." *arXiv preprint* arXiv:1907.11692, 2019. Available: `https://arxiv.org/abs/1907.11692`.
- Zhang, Xiang, Zhao, Junbo, and LeCun, Yann. "Character-level Convolutional Networks for Text Classification." In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. Available: `https://arxiv.org/abs/1509.01626`.
- Hugging Face. "Transformers: State-of-the-Art Natural Language Processing." 2020. Available: `https://huggingface.co/docs/transformers`.
- OpenAI. "ChatGPT: Optimizing Language Models for Dialogue." OpenAI, 2022. Available: `https://openai.com/research/chatgpt`.