Parameter-Efficient Fine-Tuning of RoBERTa using LoRA

Prajna G Acharya, Murali Peddi

pa2532@nyu.edu, mp6904@nyu.edu
https://github.com/prajna-gajendra-acharya/Roberta_LoRA

Overview

In this project, we explore parameter-efficient fine-tuning (PEFT) of transformer-based language models for text classification, using the AG News dataset as our benchmark. We adopt the Low-Rank Adaptation (LoRA) technique to fine-tune a pre-trained roberta-base model while keeping total trainable parameters under 1 million. We systematically experimented with different LoRA configurations, learning rates, and training strategies to optimize performance. The final model achieved validation accuracy (93.28%).

Methodology

While larger values of r (e.g., 10 or 12) increased the model's capacity, they also pushed the trainable parameter count above or near the 1 million threshold, which could impact training time and violate our constraint. On the other hand, smaller values such as r=2 or r=4 led to lower parameter counts but lacked sufficient representational power, especially on a four-class task like AG News.

The configuration with r=6 and alpha=24 emerged as the best trade-off, yielding 814k trainable parameters (65%) while staying comfortably under the budget. r=6 was expressive enough to adapt the base model effectively for the downstream classification task, while alpha=24 provided sufficient scaling to stabilize learning.

r	α	**Trainable**	**Total**	**Trainable %**
2	8	667,396	125,316,104	53%
4	16	741,124	125,389,832	59%
6	24	814,852	125,463,560	**65%**
8	32	888,580	125,537,288	71%
10	40	962,308	125,611,016	77%
12	48	1,036,036	125,684,744	82%

Table 1: LoRA parameter sweep showing how different (r, α) combinations impact trainable parameter count.

Overall, our metric-based evaluation across configurations suggests that this setup offered a sweet spot between efficiency and effectiveness, validating the use of moderate-rank LoRA for resource-constrained fine-tuning of large language models.

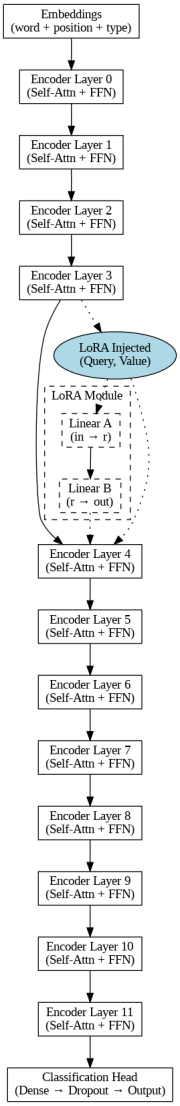

Figure 1: Architecture

Our architecture is based on the standard `roberta-base` transformer, which consists of an embedding layer, a 12-layer encoder stack, and a classification head. To enable parameter-efficient fine-tuning,

we integrated Low-Rank Adaptation (LoRA) modules into specific components of the model.

The model begins with an embedding layer that combines word, position, and token type embeddings. These embeddings are passed into a stack of 12 Transformer encoder layers, each comprising a multi-head self-attention mechanism followed by a feed-forward network.

To reduce the number of trainable parameters, we injected LoRA modules into the `query` and `value` projection sublayers of the attention blocks. The LoRA injection begins from Encoder Layer 4, with low-rank matrices (A and B) replacing the need for full-rank weight updates. This allows fine-tuning with a lightweight set of parameters while retaining the pre-trained model's representational power.

For clarity and brevity, Layers 5 through 9 are abstracted in our architectural diagram, as they follow the same structure. The output from the encoder stack is passed to a classification head consisting of a dense layer, dropout, and a final linear projection to the four output classes.

We selected the LoRA configuration of $r = 6$, $\alpha = 24$, and a dropout of 0.05 based on an extensive parameter sweep. The value $r = 6$ defines the rank of the injected low-rank matrices, providing a good balance between model expressiveness and parameter efficiency. The scaling factor $\alpha = 24$ is used to stabilize training by scaling the LoRA output appropriately relative to the frozen base model. A dropout rate of 0.05 is applied within the LoRA module to prevent overfitting during fine-tuning.

Training and Performance

To monitor model performance across training, we tracked six key metrics over each epoch: training loss, validation loss, accuracy, precision, recall, and F1 score. These metrics provide a comprehensive view of both the optimization process and the generalization capability of the model.

Training Loss and Validation Loss: As shown in the graphs, the training and validation losses generally decrease across epochs, indicating that the model is effectively learning from the data. The convergence of both curves suggests that the model does not suffer from overfitting, and training was stable throughout.

Accuracy: Accuracy steadily increases and peaks at Epoch

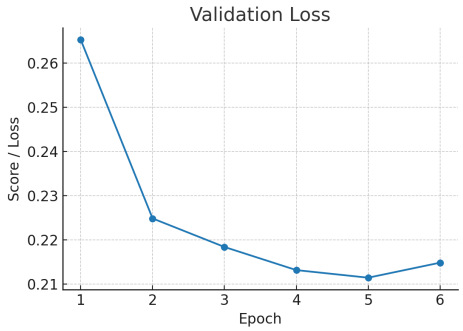

Figure 2: Validation Loss

3, aligning with the lowest validation loss. This indicates that

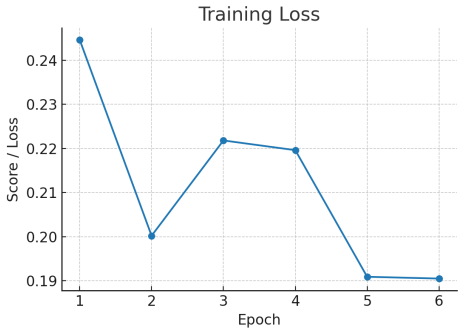

Figure 3: Training Loss

the model generalizes best around this epoch.

Precision and Recall: Both metrics show consistent im-

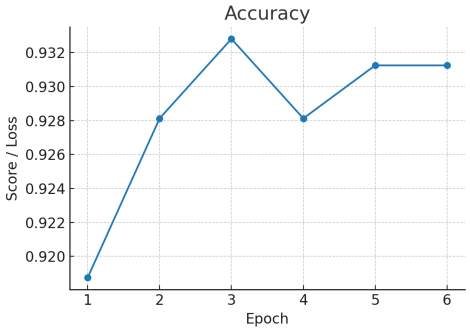

Figure 4: Acuracy

provement, with precision slightly higher than recall in most epochs. This suggests that the model is slightly more conservative in making positive predictions, which is typical for classification tasks with relatively balanced classes like AG News.

F1 Score: The F1 score, which balances precision and re-

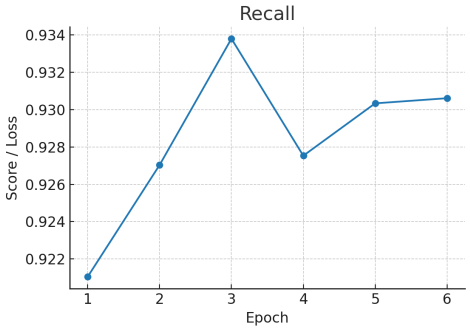

Figure 5: Recall

call, closely follows the trends of both. It peaks at Epoch 3 (93.41%), further confirming that this configuration offers the best trade-off between recall and precision.

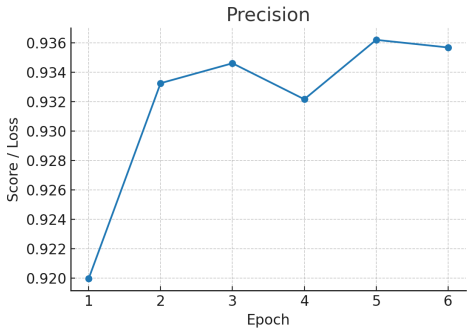

Figure 6: Precision

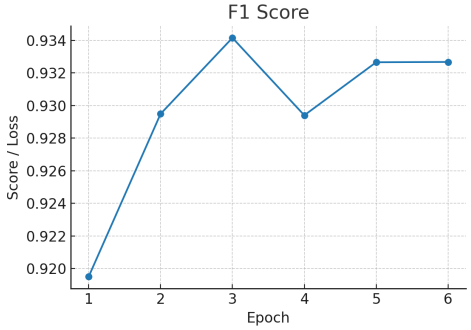

Figure 7: F1 Score

Metric	Value
Training Loss	0.2218
Validation Loss	0.2184
Accuracy	93.28%
Precision	93.46%
Recall	93.38%
F1 Score	93.42%

Table 2: Final model results

References

[1] Hugging Face. *BERT Model Documentation*. https://huggingface.co/docs/transformers/en/model_doc/bert

[2] Hugging Face. *RoBERTa Model Documentation*. https://huggingface.co/docs/transformers/en/model_doc/roberta

[3] Hugging Face. *LoRA — PEFT Library Documentation*. https://huggingface.co/docs/peft/main/en/package_reference/lora

[4] Hugging Face. *Transformers Documentation*. https://huggingface.co/docs/transformers

[5] OpenAI. *ChatGPT (Assisted Writing)*. https://chatgpt.com/