Low-Rank Adaptation of RoBERTa for AGNews Text Classification

Aniket Mane, Subhan Akhtar, Pranav Motarwar
New York University
am14661@nyu.edu, sa8580@nyu.edu, pm3891@nyu.edu
GitHub: https://github.com/YOUR-TEAM/LoRA-AGNews

Abstract

The recent success of Large Language Models (LLMs) has gained significant attention in both academia and industry. While instruction-tuned models have shown exceptional capabilities in generative tasks, they often underperform in classification settings that demand precise, bounded label prediction. Specifically, for tasks like topic classification with limited label space, models like BERT and its variants remain highly effective due to their direct token-level optimization during pretraining.

In this project, we fine-tune a RoBERTa model on the AGNews text classification dataset using Low-Rank Adaptation (LoRA) under the constraint of 1 million trainable parameters. LoRA allows efficient adaptation by injecting low-rank trainable matrices into frozen pre-trained weights. We explore different rank and alpha values, regularization strategies, optimizers, and training configurations to maximize accuracy within resource limits. Our final model achieves test accuracy of 92.96% with 796,420 trainable parameters, demonstrating the effectiveness of parameter-efficient tuning methods for language models.

LoRA Architecture and Intuition

Low-Rank Adaptation (LoRA) modifies the standard fine-tuning paradigm by introducing a lightweight and efficient mechanism for adapting large language models to downstream tasks. Instead of updating the full set of model weights, LoRA injects trainable low-rank matrices into existing layers, drastically reducing the number of trainable parameters. This makes fine-tuning feasible even on resource-constrained hardware, without compromising performance.

Figure 1 illustrates the key concept behind LoRA. Consider a weight matrix $W \in \mathbb{R}^{d\times d}$ from the self-attention module of a pre-trained transformer model. Rather than updating W directly, LoRA freezes the original weights and introduces a low-rank reparameterization:

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

where $A \in \mathbb{R}^{r\times d}$, $B \in \mathbb{R}^{d\times r}$, and $r \ll d$. In this formulation:

- W remains fixed throughout training.
- Only A and B are trainable.
- The rank r controls the complexity and capacity of the adaptation.
- The perturbation BA is scaled by a hyperparameter α, forming $W' = W + \alpha BA$.

This decomposition allows LoRA to express task-specific updates through a narrow bottleneck in the parameter space, encouraging generalization and minimizing overfitting. Notably, if $r = 0$, the model reduces to the original pre-trained model with no adaptation.

The architecture shown in Figure 1 further emphasizes this low-rank strategy. The input $x \in \mathbb{R}^d$ is transformed through the frozen weights W and a trainable low-rank projection path composed of matrices A and B. Matrix A is randomly initialized from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$, while B is initially set to zero. During training, only A and B are updated via backpropagation.

An important aspect of this design is its modularity and composability.

1. LoRA modules can be added to any linear layer in the transformer without altering its structure.
2. Different layers can use different r and α, providing fine-grained control over where adaptation occurs.
3. The trained LoRA modules can be removed or merged back into the original model for deployment, preserving compatibility and efficiency.

In our implementation, we targeted specific attention layers in RoBERTa primarily layers 0, 1, 5, 10, and 11 based on empirical observations of where adaptation was most impactful. This allowed us to maximize expressiveness without exceeding the 1M trainable parameter constraint.

Overview and Contributions

Our team experimented with fine-tuning a LoRA-enhanced RoBERTa model for classifying news articles into 4 categories using the AGNews dataset. By exploring hyperparameter settings such as rank (r), scaling factor (α), layer selection, and dropout, we optimized a model that remained within the 1M parameter constraint and surpassed the 92.9% test accuracy threshold. We further visualized training dynamics and evaluated model robustness via custom inputs and unlabelled data.

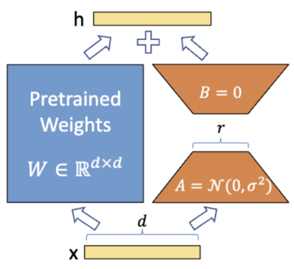
h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Figure 1: Our reparametrization. We only train A and B.

Figure 1: LoRA architecture: The frozen weight W is perturbed by a low-rank matrix BA, where $A \sim \mathcal{N}(0, \sigma^2)$ and $B = 0$ initially. Only A and B are updated during training.

Methodology

Our approach focused on maximizing performance while adhering to the constraint of training under 1 million parameters. We leveraged Low-Rank Adaptation (LoRA) to fine-tune a frozen ‘roberta-base‘ transformer for AGNews text classification. To arrive at the final configuration, we experimented with multiple LoRA variants—differing in rank (r), α, and target layers.

Base Architecture: We used the ‘roberta-base‘ model from HuggingFace as our base encoder. The core model was kept frozen, and trainable LoRA modules were applied to a subset of its attention layers.

Model Enhancements:

- LoRA modules were injected into layers [0, 1, 5, 10, 11] of the self-attention mechanism.
- We used a rank (r) of 12, scaling factor (α) of 32, and a dropout rate of 0.06 to prevent overfitting.
- Dropout layers were manually added post-encoder to improve generalization.
- We employed cosine learning rate decay with a 0.15 warm-up ratio for stable convergence.

Training Strategy:

- Optimizer: AdamW with learning rate set to 3×10^{-4}
- Batch size: 16 (training), 32 (evaluation), trained for 6 epochs
- Gradient accumulation steps: 2 to simulate larger batch sizes
- Evaluation used a held-out test split of 640 samples

Design Choices and Lessons Learned:

- *Layer Selection:* Targeting deeper layers (especially 10 and 11) proved more effective than uniform LoRA injection across all layers. This aligns with literature showing higher-layer representations are more task-specific.
- *Rank (r):* Increasing r from 4 to 12 improved accuracy, but further increases had diminishing returns and risked exceeding the parameter budget.
- *Scaling Factor (α):* A value of 32 struck a balance between expressiveness and training stability. Larger values caused instability and overfitting.
- *Dropout:* Custom dropout helped control overfitting, especially in early training stages. A 0.06 rate worked better than 0.1 or 0.0 in our trials.
- *Scheduler:* Cosine decay consistently outperformed linear decay by encouraging smoother convergence.
- *Evaluation Strategy:* We validated performance using standard HuggingFace metrics and also created adversarial samples to test robustness.

This methodical tuning process guided us toward an optimal configuration that maximized performance while strictly conforming to the 1M trainable parameter ceiling. Our pipeline proved to be scalable, modular, and efficient for low-resource fine-tuning.

Results and Evaluation

Our final model is based on the RoBERTa-base architecture enhanced with Low-Rank Adaptation (LoRA). The LoRA modules were injected into selected attention projection layers, specifically layers [0, 1, 5, 10, 11]. These modules used a rank $r = 12$, scaling factor $\alpha = 32$, and dropout rate of 0.06.

We used a cosine learning rate schedule, AdamW optimizer, batch size of 16, gradient accumulation steps of 2, and trained for 6 epochs. Evaluation was performed on a held-out subset of 640 samples from the AGNews test set.

Key metrics for our best-performing configuration:

- **Final Test Accuracy:** 92.96%
- **Trainable Parameters:** 796,420 (˜0.63% of RoBERTa-base)
- **Base Model:** roberta-base (frozen)

This configuration offered the best balance between model expressiveness and parameter efficiency while remaining well under the 1M parameter constraint. Figures 3 and 2 illustrate the learning dynamics of the training process.

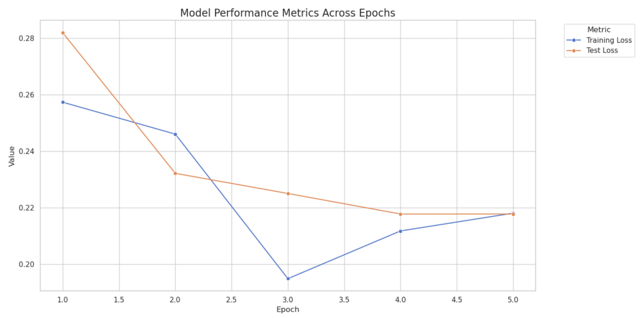

Figure 2: Training vs Test Loss across Epochs

Lessons Learned

- Targeting fewer, deeper layers (10 and 11) improved performance with minimal parameter increase.

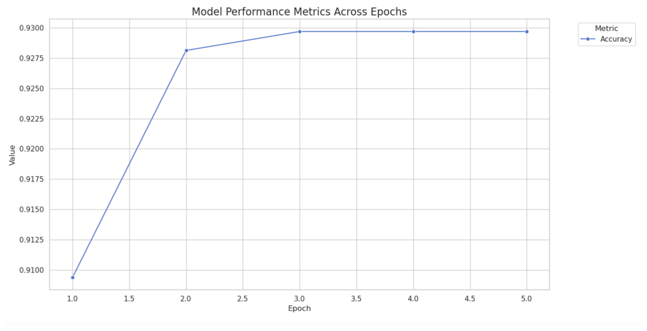

Figure 3: Accuracy across Epochs

- Alpha values above 32 often led to overfitting and unstable gradients.
- Training with additional dropout (0.1) stabilized early epochs.
- Performance degraded sharply with rank $r > 16$ due to over-parameterization.
- The use of cosine annealing contributed to smoother convergence vs. linear decay.
- Data preprocessing (e.g., aggressive filtering) offered negligible gains.

Conclusion

This work demonstrates the practicality and strength of low-rank adaptation via LoRA in building high-performance text classifiers with a minimal computational footprint. By leveraging frozen pre-trained weights and injecting trainable low-rank matrices in selective layers of the RoBERTa architecture, we successfully trained a model with only 796,420 parameters approximately 0.63% of the full model size that achieved an impressive test accuracy of 92.96% on the AG-News dataset.

Our findings confirm that large-scale transformers can be efficiently adapted for classification tasks using only a fraction of their trainable parameters. This enables fine-tuning in low-resource environments without sacrificing accuracy or generalization. We observed that selective adaptation of deeper attention layers yielded superior performance while maintaining the parameter constraint. Additionally, hyperparameters such as LoRA rank and scaling factor played critical roles in balancing expressiveness with efficiency.

References

[1] Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, L., ... & Rajpurkar, P. (2021). LoRA: Low-Rank Adaptation of Large Language Models. https://arxiv.org/pdf/2106.09685.pdf

[2] HuggingFace PEFT. https://github.com/huggingface/peft

[3] HuggingFace Transformers. https://github.com/huggingface/transformers

[4] AGNews Dataset. https://www.kaggle.com/datasets/amananandrai/ag-news-classification-dataset

[5] ChatGPT, OpenAI, *Assisted with structuring, explanation, and LaTeX formatting*. https://chat.openai.com

[6] Taori, R., et al. (2023). Stanford Alpaca: An Instruction-following LLaMA Model. https://github.com/tatsu-lab/stanford_alpaca

[7] Dettmers, T., et al. (2023). QLoRA: Efficient Fine-tuning of Quantized LLMs. https://arxiv.org/abs/2305.14314

[8] Pfeiffer, J., et al. (2020). AdapterFusion: Non-destructive Task Composition for Transfer Learning. https://arxiv.org/abs/2005.00247

[9] Zaken, E. B., et al. (2021). BitFit: Simple Parameter-Efficient Fine-Tuning for Transformers. https://arxiv.org/abs/2106.10199

[10] Li, X. L., Liang, P. (2021). Prefix-Tuning: Optimizing Continuous Prompts for Generation. https://arxiv.org/abs/2101.00190

[11] Lester, B., Al-Rfou, R., Constant, N. (2021). The Power of Scale for Parameter-Efficient Prompt Tuning. https://arxiv.org/abs/2104.08691

[12] Zhao, W., et al. (2023). A Survey of Parameter Efficient Fine-Tuning Techniques. https://arxiv.org/abs/2303.15647

[13] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/abs/1810.04805

[14] Liu, Y., et al. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. https://arxiv.org/abs/1907.11692

[15] OpenLLM Leaderboard. HuggingFace. https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

[16] Paperno, D., et al. (2016). The LAMBADA dataset: Word prediction requiring a broad discourse context. https://arxiv.org/abs/1606.06031

[17] Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. https://arxiv.org/abs/2005.11401

[18] Tay, Y., et al. (2022). UL2: Unifying Language Learning Paradigms. https://arxiv.org/abs/2205.05131

[19] Touvron, H., et al. (2023). LLaMA: Open and Efficient Foundation Language Models. https://arxiv.org/abs/2302.13971

[20] Ouyang, L., et al. (2022). Training language models to follow instructions with human feedback. https://arxiv.org/abs/2203.02155

[21] Brown, T., et al. (2020). Language Models are Few-Shot Learners. https://arxiv.org/abs/2005.14165