Deep Learning Finetuning with LoRA Project

Jesse Noppe-Brandon

New York University
CS 6923
jn2934@nyu.edu

Abstract

In this project, we adapt a frozen `RoBERTa-base` model to the AGNEWS text classification task under a strict budget of 1 million trainable parameters by leveraging Low-Rank Adaptation (LoRA). We systematically explore the LoRA design space—varying the attention modules targeted (query, key, value, and their combinations), adapter rank r, scaling factor α, and dropout rate—alongside choices of optimizer (SGD, RMSProp, AdamW), learning rate schedulers, and training duration. Our experiments reveal that confining LoRA to the self-attention *value* projection with $r = 22$, $\alpha = 44$, and dropout = 0.10 delivers the greatest accuracy gains. Coupled with the AdamW optimizer at a peak learning rate of 5×10^{-5} this configuration achieves a peak validation accuracy of **91.41%** within 1200 steps. These results demonstrate that careful allocation of limited adapter capacity can approach full fine-tuning performance while maintaining a minimal trainable footprint. Our implementation builds on the Hugging Face `Transformers` and `PEFT` libraries, providing a reproducible blueprint for parameter-efficient adaptation of large language models.

https://github.com/jessenb16/Finetuning-with-LoRA .

Introduction

Fine-tuning large pre-trained language models for downstream tasks has become a standard approach to achieve state-of-the-art performance in natural language processing. However, full fine-tuning can be prohibitively expensive in terms of both compute and storage when model sizes reach hundreds of millions—or even billions—of parameters. Low-Rank Adaptation (LoRA) (Hu et al. 2021) offers a parameter-efficient alternative: it freezes the bulk of the pre-trained model weights and injects small, trainable low-rank matrices into selected weight projections. By controlling the rank r of these update matrices and a scaling factor α, LoRA can achieve comparable performance to full fine-tuning while training only a small fraction of the model parameters.

In this work, we apply LoRA to a RoBERTa-base encoder on the AGNEWS text classification dataset under a strict budget of 1 million trainable parameters. We begin with a "base" LoRA configuration chosen to maximize capacity under the budget:

- r set to the largest value yielding $\leq 10^6$ trainable parameters,
- $\alpha = 2r$ (a commonly recommended ratio (Raschka 2023)),
- dropout $= 0.05$.

For training, we adopt the Hugging Face default for AdamW (`learning_rate` $= 5 \times 10^{-5}$), with 1,200 total steps, 100 warmup steps, step-based evaluation every 100 steps, and a batch size of 16 per device.

We then conduct a sequence of controlled experiments to identify the most effective LoRA setup:

1. **Adapter Scope:** sweep over targeting the `query`, `key`, and `value` projections (and their pairwise and triplet combinations). We found that `value` alone reached top accuracy fastest and with lowest loss.
2. **Dropout:** with `value` adapters fixed, compare dropout rates $\{0.05, 0.10, 0.15\}$, selecting 0.10 as optimal.
3. **Scaling Factor α:** holding r and dropout constant, test $\alpha \in \{0.5r,\ 1r,\ 2r\}$, confirming $2r$ is best.
4. **Rank r:** evaluate slightly smaller and larger r around the 1M-parameter boundary; the largest feasible r gave the highest accuracy.
5. **Learning Rate:** compare 2×10^{-5}, 3×10^{-5}, 5×10^{-5}, with 5×10^{-5} yielding the best performance.
6. **Scheduler & Regularization:** explore cosine and cosine-with-restarts schedules and light weight decay; none substantially improved over the base setup.

Through this systematic exploration, we identify that a LoRA adapter on the `value` projection with $r = 22$, $\alpha = 44$, dropout = 0.10, trained via AdamW at 5×10^{-5} for 1,200 steps, attains a peak validation accuracy of **91.41%**. These findings demonstrate that carefully constrained LoRA fine-tuning can match the performance of full fine-tuning while adhering to strict parameter budgets. The following sections detail our methodology, experimental results, and insights for parameter-efficient adaptation of large language models.

Methodology

Baseline Training Configuration

We began by establishing a robust baseline for training the LoRA-augmented RoBERTa model, focusing on optimizer

choice and total training steps. Our goal was to identify settings that reliably converge without over- or under-training, before exploring adapter-specific hyperparameters.

Optimizer comparison. We compared three optimizers—SGD, RMSProp, and AdamW. While RMSProp converged very slowly and SGD plateaued below competitive accuracy, AdamW consistently delivered the highest validation accuracy and most stable loss curves. Consequently, all subsequent experiments use AdamW.

Step count sweep. To determine an appropriate number of training steps, we ran short experiments at 200, 400, 800, 1200, and 2400 steps. We observed rapid improvements through 1200 steps but negligible gains or slight over-fitting beyond that point. Therefore, we fixed the total training budget at *1200 steps* for all downstream experiments.

Other base settings. Unless otherwise noted, every run uses:

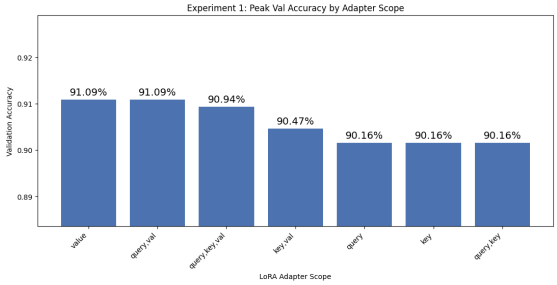
- **Learning rate:** 5×10^{-5} (the Hugging Face–recommended default for AdamW)
- **Warmup steps:** 100 ($\approx 8\%$ of total steps)
- **Evaluation strategy:** step-based every 100 steps
- **Batch size:** 16 per device for both training and evaluation
- **Data loader workers:** 4 parallel workers

These baseline hyperparameters form the foundation for all subsequent LoRA configuration sweeps.

Experiment 1: LoRA Adapter Scope

We first set the adapter rank r to the largest value that kept the total trainable parameters under one million, and chose the scaling factor $\alpha = 2r$. With all other hyperparameters held constant, we evaluated all seven possible subsets of the self-attention projections—`query`, `key`, and `value`—as LoRA adapter targets. All configurations yielded 999172 trainable parameters except the triplet combination (`[query,key,value]`), which had 980740 parameters; these represent the maximum budgets achievable given our chosen r values.

As shown in Figure 1, both `value` and `query,value` reached a peak validation accuracy of 91.09%. However, the `value`-only adapter hit that level 100 steps earlier and exhibited the lowest evaluation loss among all combinations. Hence, we selected `value` as our adapter scope for all subsequent experiments.



	step	eval_loss	r	lora_alpha
value	1000	0.297	22	44
query,val	1100	0.302	11	22
query,key,val	1200	0.305	7	14
key,val	900	0.312	11	22
query	1100	0.311	22	44
key	800	0.321	22	44
query,key	1000	0.321	11	22

Figure 1: Experiment 1: Peak validation accuracy by LoRA adapter scope. The bar chart displays the highest accuracy attained by each scope, and the table below lists the corresponding step, evaluation loss, r, and α.

Rationale for the `value` Projection The self-attention `value` projection determines what each attention head emits to the rest of the network, making it especially effective for task-specific adaptation. By focusing the entire adapter capacity (rank $r = 22$) on this single projection, we maximize expressive power under the 1 million parameter budget, enabling faster convergence and lower loss compared to multi-module adapters.

Experiment 2: Dropout Rate

With the adapter scope fixed to `value` and hyperparameters $r = 22$, $\alpha = 44$, we evaluated three dropout rates $\{0.05, 0.10, 0.15\}$.

Figure 2 plots the validation curves for each dropout setting. The 0.10 dropout model achieved a peak accuracy of 91.41%, improving on the previous best (91.09% at dropout=0.05) by 0.32 percentage points. Dropout=0.15 performed the worst, indicating that moderate regularization is optimal under our parameter budget.

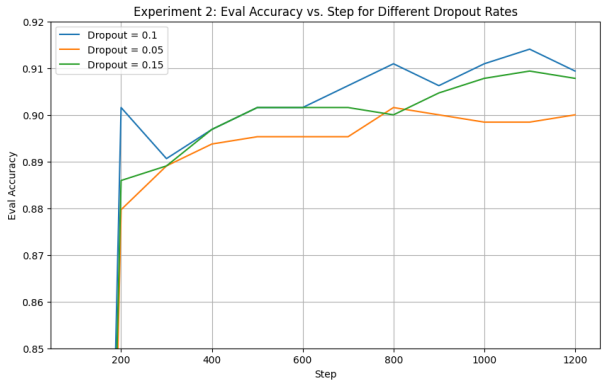


Figure 2: Experiment 2: Validation accuracy vs. training step for different LoRA dropout rates (value-only adapter). Dropout=0.10 reaches the highest peak (91.41%), while 0.15 lags behind.

Discussion A dropout rate of 0.10 strikes the right balance between preventing over-fitting and preserving signal strength in the low-rank adapters. Lower dropout (0.05) under-regularizes, while higher dropout (0.15) removes too much information, slowing adaptation.

Experiment 3: Scaling Factor α

We tested three scaling factors $\alpha \in \{0.5r, 1r, 2r\} = \{11, 22, 44\}$ on the `value`-only adapter with $r = 22$ and dropout=0.10. Figure 3 shows the validation accuracy curves for each setting.

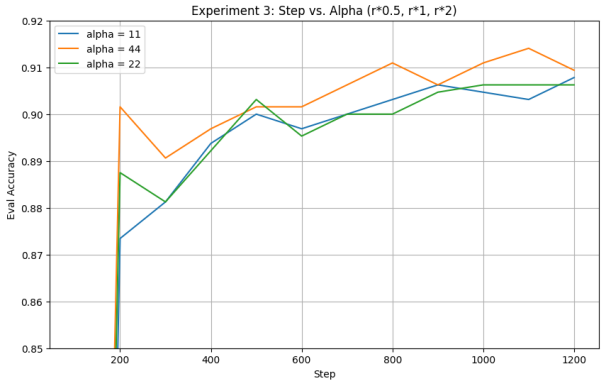


Figure 3: Validation accuracy vs. training step for scaling factors $\alpha = 0.5r$, $1r$, and $2r$ (value-only adapter, $r = 22$). $\alpha = 2r$ reaches the highest peak accuracy.

Discussion We evaluated three scaling factors—$\alpha = 0.5r$, $\alpha = 1r$, and $\alpha = 2r$—and observed that the two smaller values ($0.5r$ and $1r$) yielded similar, moderate performance. In contrast, $\alpha = 2r$ consistently delivered a higher peak validation accuracy, confirming that amplifying the low-rank updates by a factor of two provides sufficient expressivity without destabilizing the frozen weights. These results endorse $\alpha = 2r$ as a robust default for LoRA.

Experiment 4: Adapter Rank r

To assess the impact of adapter capacity under a fixed scope (`value` only) and scaling ($\alpha = 2r$), we tested three ranks:

$$r \in \{6,\ 11,\ 22\},$$

corresponding to approximately 704 K, 796 K, and 999 K trainable parameters, respectively.

Figure 4 shows the validation accuracy curves for each r. The highest-rank adapter ($r = 22$, 999172 params) yields the best performance, while the smallest adapter ($r = 6$, 704260 params) performs worst. This monotonic improvement confirms that—once the optimal projection scope is chosen—allocating as many parameters as possible to the low-rank adapter maximizes accuracy.

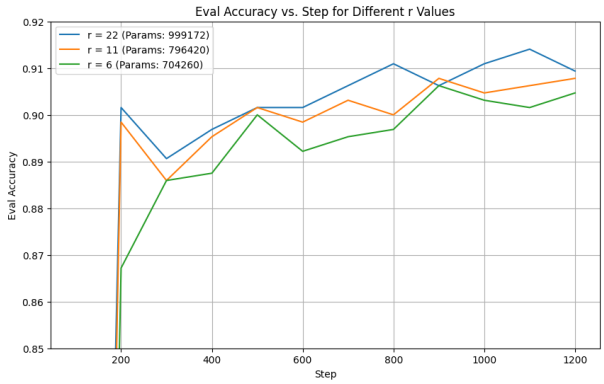


Figure 4: Experiment 4: Validation accuracy vs. training step for adapter ranks $r = 6$, 11, 22. Higher r (more trainable parameters) produces consistently better accuracy under the same training budget.

Discussion These results demonstrate that, after fixing the adapter scope to the `value` projection, model performance scales with the available parameter budget. By minimizing the number of adapter modules and channeling capacity into a single projection, we can maximize r and thus fully leverage our 1 million parameter limit for optimal adaptation.

Experiment 5: Learning Rate

We evaluated three peak learning rates $\{2 \times 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}\}$ under the fixed configuration (`value` adapter, $r = 22$, $\alpha = 44$, dropout=0.10). Longer runs beyond 1200 steps showed no significant improvements, confirming that 1200 steps sufficed.

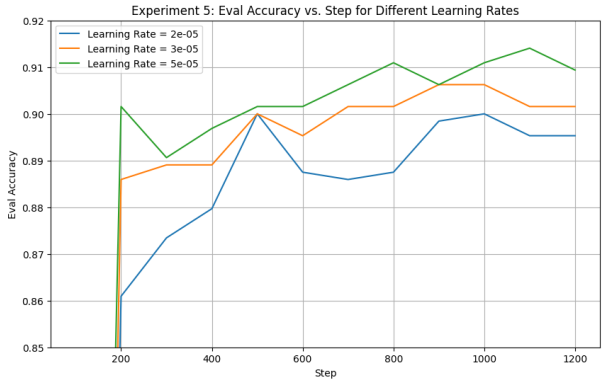


Figure 5: Experiment 5: Validation accuracy vs. training step for different learning rates. Higher learning rates within this range (up to 5×10^{-5}) yield consistently better peak accuracy.

Results and Discussion The curves in Figure 5 exhibit a clear monotonic trend:

- 2×10^{-5} peaks around 89.5%,
- 3×10^{-5} around 90.2%,
- 5×10^{-5} around 91.4%.

A higher learning rate accelerates adaptation of the low-rank matrices without destabilizing the frozen base weights, up to the tested maximum. Beyond 1200 steps, none of the rates showed further gains, validating our earlier step-count experiments.

Experiment 6: Scheduler and Regularization

In our final set of experiments, we explored two additional training axes—learning-rate scheduling and weight decay—to determine if they could further improve performance beyond the 91.41% peak.

Learning-Rate Schedules We compared:

- **Cosine decay:** a smooth, monotonic reduction from the initial learning rate to zero over 2400 steps,
- **Cosine with restarts:** two cosine cycles of 1200 steps each,

Neither of these schedules produced validation accuracy above 91.41%, and in fact took longer to reach similar accuracies than the default Linear scheduler.

Weight Decay We also tested light regularization via weight decay values $\{0.0, 0.01, 0.1\}$. Across these settings, validation accuracy never exceeded 91.41%.

Discussion These additional experiments may demonstrate that—once adapter scope, rank, scaling, dropout, and learning rate are well tuned—further tweaks to schedule shape and weight decay don't improve results under the 1 M parameter constraint.

Conclusion

In this study, we have shown that careful allocation of limited trainable parameters via LoRA can yield strong performance on AGNEWS, achieving 91.41% validation accuracy with only 1 million parameters. Two central insights emerged:

Value Projection Is Paramount Adapting the self-attention `value` projection alone delivered the largest accuracy gains. This projection controls what each attention head emits to the subsequent layers, making it the most direct lever for injecting task-specific information. By concentrating the adapter's capacity on `value`, we enabled the model to re-interpret the frozen representations most effectively.

Maximize Adapter Rank Once the optimal scope was identified, increasing the adapter rank r to the largest feasible value (22, corresponding to 999172 parameters) produced monotonic improvements in accuracy. This demonstrates that—under a hard parameter budget—allocating as much capacity as possible to the low-rank updates maximizes expressivity and performance.

Recipe Our final, streamlined recipe—LoRA on `value` with $r = 22$, $\alpha = 44$, dropout=0.10, AdamW at 5×10^{-5} for 1200 steps—provides a reproducible blueprint for parameter-efficient adaptation of transformer models.

Future improvements Future work might extend LoRA to feed-forward layers, incorporate lightweight data augmentation, or explore quantization-aware protocols to further push the accuracy–efficiency frontier.

A full factorial sweep across all module combinations would further ensure finding the global optimum, though at higher compute cost.

Final Results

After extensive experimentation, the best model configuration was achieved with the following parameters:

Parameter	Value
LoRA Configuration	
Base Model	`roberta-base`
Adapter Scope	`value`
Rank (r)	22
Scaling Factor (α)	44
Dropout	0.10
Trainable Parameters	999,172
Training Parameters	
Optimizer	AdamW (`adamw_torch`)
Learning Rate	5×10^{-5}
Warmup Steps	100
Total Steps	1,200
Eval Strategy	steps (every 100)
Train Batch Size	16
Eval Batch Size	64
DataLoader Workers	4
Final Performance	
Training Loss	0.2993
Validation Loss	0.2965
Peak Validation Accuracy	**91.41%**

Table 1: Final LoRA-augmented RoBERTa configuration and performance on AGNEWS under the 1 million-parameter constraint.

References

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685.

Raschka, S. 2023. Practical Tips for Finetuning LLMs Using LoRA (Low-Rank Adaptation). https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms. Accessed: 2025-04-20.