

Exploring Low-Rank Adaptation for Efficient GPT-2 Fine-Tuning

Stanford CS224N Default Project

Shengzhe (Andy) Zhang

Department of Civil & Environmental Engineering
Stanford University
andyysz@stanford.edu

Yi Lu

Department of Civil & Environmental Engineering
Stanford University
yilu1@stanford.edu

Abstract

Full fine-tuning remains the standard recipe for adapting pretrained language models, but it is expensive in both memory and optimization cost. We study whether Low-Rank Adaptation (LoRA) can recover most of the benefits of full GPT-2 fine-tuning on the Stanford CS224N default project while substantially reducing adaptation cost. Using a from-scratch GPT-2 implementation initialized with pretrained weights, we evaluate LoRA on two downstream settings with different inductive demands: cloze-style paraphrase detection on Quora Question Pairs (QQP) and sonnet generation. We vary adapter placement, rank, and optimization hyperparameters to characterize the efficiency–performance results.

On paraphrase detection, full fine-tuning reaches 0.899 development accuracy and 0.892 development F1, with 77.50 minutes per epoch and 4.699 GB peak memory. The best LoRA configuration—joint attention+MLP adaptation with rank $r = 8$ and learning rate 1×10^{-4} —achieves 0.894 development accuracy and 0.887 development F1 while training only 1.05% of parameters comparing to full fine-tuning, reducing peak memory to 3.557 GB and epoch time to 52.21 minutes. This leaves only a 0.005 absolute accuracy gap while cutting epoch time by 32.6% and peak memory by 24.3%.

On sonnet generation, the picture is less stable. The baseline reaches 41.182 chrF, and the strongest completed LoRA run matches this score under an attention-only configuration with rank $r = 4$ and learning rate 5×10^{-5} . However, generation quality is substantially more sensitive to placement and optimization choices than paraphrase performance. Overall, our results show that LoRA is highly competitive for structured semantic prediction and can recover nearly all of the gains of full fine-tuning at a small fraction of the cost, whereas its behavior on open-ended generation is markedly more configuration-sensitive.

1 Key Information to include

- Mentor: Alisa Levin
- External Collaborators (if you have any): No
- Sharing project: No
- Late Days (if you are using any): 3 (None of us used late days before)

2 Introduction

Fine-tuning pretrained language models is highly effective, but updating the full parameter set is expensive and resource-wasting. As model scale grows, the cost of storing gradients, optimizer states, and dense updates directly limits experimental throughput and makes systematic ablation increasingly difficult. Even for GPT-2, full-model adaptation is expensive enough that parameter-efficient alternatives are practically valuable.

Parameter-efficient fine-tuning (PEFT) methods aim to preserve most of the benefits of full adaptation while updating only a small subset of parameters. Among them, Low-Rank Adaptation (LoRA) [4] is especially attractive because it freezes the pretrained backbone and adds trainable low-rank residuals into existing linear layers to reduce training pressure. This leads to a concrete empirical question: how much downstream performance can be retained when only a tiny fraction of model parameters is trainable?

We study this question in the setting of the Stanford CS224N default final project using GPT-2 as a shared backbone across two tasks with different adaptation requirements. We first completed the required GPT-2 implementation and validated it on the default sentiment-analysis tasks. We then used this implementation for two extensions: cloze-style paraphrase detection on Quora Question Pairs (QQP), where GPT-2 predicts *yes* or *no* for a question pair; and sonnet generation, where the model autoregressively continues a poem from its opening lines.

These tasks expose different aspects of adaptation. Paraphrase detection, although cast in generative form, is fundamentally a constrained semantic decision problem. Sonnet generation is open-ended and requires coherent long-range token prediction under stylistic constraints. This contrast allows us to ask not only whether LoRA works, but where it works most reliably. Our central question is:

Can LoRA recover most of the performance of full GPT-2 fine-tuning while substantially reducing trainable parameters, memory usage, and training time, and does this trade-off performance differ between different tasks (structured semantic prediction and open-ended generation)?

Our main finding is a clear task asymmetry. On paraphrase detection, LoRA nearly closes the gap to full fine-tuning while substantially improving efficiency. On sonnet generation, LoRA can match the baseline, but the result is much less robust to adapter placement and hyperparameter choice.

3 Related Work

Our study connects three lines of work: autoregressive Transformer language models, paraphrase modeling, and parameter-efficient fine-tuning.

Autoregressive Transformer language models. The Transformer established self-attention as the dominant sequence-modeling primitive [1]. GPT-style decoder-only models then showed that large-scale next-token prediction produces broadly transferable representations [2]. GPT-2 is especially relevant here because it supports both discriminative reformulations and open-ended generation within a single decoder-only architecture [2].

Paraphrase detection. Paraphrase detection is a standard benchmark for semantic equivalence beyond lexical overlap. QQP, also used in GLUE, is one of the most common datasets for this setting [3]. In the CS224N default project, QQP is reformulated as cloze-style generation, enabling direct comparison of adaptation strategies without introducing a separate classification architecture.

Parameter-efficient fine-tuning. LoRA freezes pretrained weights and learns low-rank residual updates for selected linear transformations [4]. Its core intuition is that task-specific adaptation often lies in a low-dimensional subspace, making dense full-model updates unnecessarily expensive. Our work does not propose a new PEFT method; rather, it studies how LoRA behaves in a controlled student-project setting across two downstream tasks with distinct demands, emphasizing adapter placement, rank, parameter setting, and optimization sensitivity.

4 Approach

4.1 Backbone model

Our backbone is GPT-2 base, implemented from scratch following the CS224N default project and initialized with pretrained GPT-2 weights. The model contains token embeddings, positional embeddings, 12 Transformer decoder blocks, masked multi-head self-attention, position-wise feed-forward networks (MLPs), residual connections, and layer normalization. Self-attention is computed as

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \quad (1)$$

As a baseline, full fine-tuning updates all pretrained weights:

$$W \leftarrow W - \eta \nabla_W \mathcal{L}, \quad (2)$$

where W is a trainable parameter matrix, η is the learning rate, and $\mathcal{L}$ is the task loss.

4.2 Base sentiment classification

In the required first stage of the default project, GPT-2 is adapted to sentence-level sentiment analysis using the final token representation. The last token embedding is passed through dropout and a linear layer to predict sentiment labels. We evaluate two modes: *last-linear-layer*, which freezes GPT-2 and trains only the classifier head, and *full-model*, which updates the entire pretrained model. These runs serve as implementation validation.

4.3 Cloze-style paraphrase detection

Although QQP is naturally a binary classification task, the default project casts it as cloze-style generation. Given two questions q_1 and q_2 , GPT-2 predicts the next token corresponding to *yes* or *no*. Let h_T denote the hidden state at the answer position. GPT-2 maps this state to a vocabulary distribution:

$$p(y \mid q_1, q_2) = \text{Softmax}(W_{\text{vocab}}h_T + b). \quad (3)$$

Prediction is obtained by comparing the probabilities assigned to the answer tokens. This preserves the decoder-only interface while enabling direct evaluation with development accuracy and F1.

4.4 Sonnet generation

For sonnet generation, GPT-2 conditions on the opening lines of a poem and generates the continuation autoregressively. Unlike paraphrase detection, this task is open-ended and sequence-level. It therefore tests not only local token prediction but also fluency, stylistic consistency, and longer-range planning. We evaluate generation quality using chrF [7], a character-level n -gram overlap metric.

4.5 Low-Rank Adaptation

To reduce adaptation cost, we replace dense updates with LoRA. Instead of learning a full update for a weight matrix W , LoRA freezes W and learns a low-rank residual:

$$W' = W + \Delta W, \quad \Delta W = \frac{\alpha}{r}BA, \quad (4)$$

where $A \in \mathbb{R}^{r \times d}$ and $B \in \mathbb{R}^{d' \times r}$ are trainable matrices, r is the rank, and α is a scaling factor [4]. In all LoRA experiments, pretrained backbone weights remain frozen.

4.6 Experimental factors

We vary three factors.

Target modules. For paraphrase detection and sonnet generation, we compare:

- (1) **Attention-only LoRA**, applied to attention projections;
- (2) **MLP-only LoRA**, applied to feed-forward projections;
- (3) **Attention+MLP LoRA**, applied jointly to both.

For sonnet generation, we use the same naming convention: attention-only, mlp-only, and attention+mlp.

Rank. We sweep

$$r \in \{4, 8, 16\}. \quad (5)$$

Learning rate. For the strongest paraphrase family (attention+MLP with $r = 8$), we additionally sweep

$$\{1 \times 10^{-5}, 2 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}\}.$$

For sonnet generation, we also explored additional variants involving learning rate, dropout, alpha, extended training, and decoding temperature.

5 Experiments

5.1 Data and evaluation

We use the datasets provided in the CS224N default project. For sentiment analysis, we evaluate on Stanford Sentiment Treebank (SST) and CFIMDB. SST is a five-way sentence-level movie-review dataset with 8,544 train, 1,101 dev, and 2,210 test examples [5, 6]. CFIMDB is a binary review dataset with 1,701 train, 245 dev, and 488 test examples [6].

For paraphrase detection, we use the provided QQP subset with 141,506 training examples, 20,215 development examples, and 40,431 test examples [6]. Each example contains two questions and a binary label, reformulated as cloze-style prediction of *yes* or *no*. For sonnet generation, we use the Shakespeare sonnet dataset in the starter code, consisting of 143 training sonnets and 12 held-out sonnets used for development evaluation [6]. All tokenization follows the official GPT-2 byte-pair encoding pipeline.

For sentiment tasks, we report development accuracy. For paraphrase detection, we report development accuracy and development F1, along with trainable parameter count, percentage of trainable parameters, peak GPU memory, and average epoch time. For sonnet generation, we report development chrF and elapsed runtime. All main comparisons are made on development results because model selection was performed on the dev split within the project compute budget.

5.2 Experimental details

All experiments use GPT-2 base. The final full-fine-tuning paraphrase run uses AdamW, learning rate 1×10^{-5} , batch size 8, and 10 epochs. Our LoRA runs keep the backbone, task formulation, batch size, and training horizon fixed while varying insertion location, rank, and learning rate. Across paraphrase LoRA runs, the scaling factor is $\alpha = 16$ and LoRA dropout is 0.0 unless otherwise stated in the sonnet experiment tags.

5.3 Base-task validation: sentiment analysis

Before moving to the LoRA study, we validated the required base implementation on the two sentiment-analysis tasks. Table 1 summarizes the results.

These results match the expected behavior of the default project: full-model fine-tuning clearly outperforms training only the final linear layer on both datasets. On SST, our full-model result of 0.509 is close to the handout baseline of 0.513, while on CFIMDB our full-model result of 0.976 matches the handout baseline exactly. This provides a sanity check that the GPT-2 implementation, pretrained-weight loading, and optimization pipeline are functioning correctly.

Fine-tuning Mode	SST Dev Acc.	CFIMDB Dev Acc.
Last linear layer	0.456	0.857
Full model	0.509	0.976

Table 1: Development accuracy on the two required sentiment-analysis tasks.

Family	Setting	Dev Acc.	Dev F1
Full FT	lr = 1e-5	0.899	0.892
Attention-only	r = 4, lr = 1e-5	0.854	0.845
Attention-only	r = 8, lr = 1e-5	0.855	0.847
Attention-only	r = 16, lr = 1e-5	0.854	0.845
MLP-only	r = 4, lr = 1e-5	0.862	0.856
MLP-only	r = 8, lr = 1e-5	0.866	0.859
MLP-only	r = 16, lr = 1e-5	0.863	0.855
Attention+MLP	r = 4, lr = 1e-5	0.868	0.860
Attention+MLP	r = 8, lr = 1e-5	0.870	0.863
Attention+MLP	r = 16, lr = 1e-5	0.865	0.855
Attention+MLP	r = 8, lr = 2e-5	0.882	0.874
Attention+MLP	r = 8, lr = 5e-5	0.891	0.883
Attention+MLP	r = 8, lr = 1e-4	0.894	0.887
Attention+MLP	r = 8, lr = 2e-4	0.888	0.881

Table 2: Paraphrase detection performance across full fine-tuning and LoRA configurations.

5.4 Results: paraphrase detection

Table 2 summarizes the completed paraphrase experiments; full performance and efficiency statistics appear in Appendix Tables 3 and 4.

The paraphrase results are consistent. Full fine-tuning remains best overall at 0.899 development accuracy and 0.892 development F1, but the strongest LoRA configuration—joint attention+MLP adaptation with rank $r = 8$ and learning rate 1×10^{-4} —reaches 0.894 accuracy and 0.887 F1 while training only 1.05% of parameters. The absolute accuracy gap is therefore only 0.005, while peak memory drops from 4.699 GB to 3.557 GB and epoch time drops from 77.50 to 52.21 minutes.

Three patterns are especially clear. First, adapter placement matters more than raw adapter size. Across the rank sweep, the family ordering is stable:

$$\text{attention+MLP} > \text{MLP-only} \gg \text{attention-only}.$$

This suggests that cloze-style paraphrase detection benefits from adapting both contextual interaction and feed-forward feature transformation.

Second, rank effects are non-monotonic. Within each family, moving from $r = 4$ to $r = 8$ improves performance, but moving from $r = 8$ to $r = 16$ does not, it actually decreases. Larger adapters therefore do not automatically help under a fixed training budget.

Third, optimization quality matters substantially. Holding the family and rank fixed (attention+MLP, $r = 8$), increasing the learning rate from 1×10^{-5} to 1×10^{-4} improves development accuracy from 0.870 to 0.894. This improvement is larger than many gains obtained by changing rank or applying LoRA to different layer.

The training curves reinforce the same picture. Full fine-tuning reaches the highest ceiling, but the strongest all-target LoRA runs rapidly approach a nearby plateau, whereas attention-only and MLP-only LoRA converge to visibly lower ceilings (Appendix Figure 1).

5.5 Results: sonnet generation

Detailed sonnet-generation results appear in Appendix Table 5. The baseline reaches 41.182 chrF, and the strongest completed LoRA run—attention-only with rank $r = 4$ and learning rate 5×10^{-5} —

matches this score. This shows that LoRA can recover baseline generation quality in at least some settings without dense full-model adaptation.

At the same time, the sonnet results are much less regular than the paraphrase results. Nearby configurations often differ sharply, and broader adapter coverage does not reliably improve chrF. In particular, attention-only adaptation is consistently stronger than mlp-only or attention+mlp adaptation, the opposite of what we observe on paraphrase detection. This pattern is summarized in Appendix Figure 5, which compares baseline generation with attention-only, MLP-only, and attention+MLP LoRA across rank settings.

Generation is also more sensitive to optimization and decoding choices. Small changes in learning rate, dropout, training length, or decoding temperature often produce noticeable drops in chrF. This suggests a rougher and narrower operating regime for LoRA in open-ended generation than in constrained semantic prediction. The same instability appears in Appendix Figure 6, where score does not improve monotonically with additional runtime across LoRA configurations.

We emphasize, however, that the sonnet findings should be interpreted more cautiously than the paraphrase findings. The development set is small and chrF is only a partial proxy for poetic quality. The main conclusion is therefore comparative rather than absolute: LoRA can be competitive for generation, but the configuration space is less forgiving.

6 Analysis

6.1 LoRA nearly closes the gap on structured semantic prediction

The strongest result of the project is that LoRA is highly effective for cloze-style paraphrase detection. The best completed LoRA model reaches 0.894 development accuracy versus 0.899 for full fine-tuning, leaving only a 0.005 gap while updating only 1.05% of parameters. This supports the view that, for a structured semantic decision problem, most useful downstream adaptation can be captured in a compact low-rank subspace.

The efficiency gains are practically meaningful rather than merely notational. Relative to full fine-tuning, the best LoRA model reduces epoch time by 32.6% and peak memory by 24.3%. The resulting trade-off is summarized in Appendix Figure 4: several LoRA configurations approach the full fine-tuning reference at substantially lower cost.

6.2 Placement is more informative than parameter count alone

A central lesson from the paraphrase study is that where LoRA is inserted matters more than how many trainable parameters it adds. MLP-only LoRA at $r = 8$ outperforms attention-only LoRA at $r = 16$, despite the latter having comparable order-of-magnitude adaptation scale. This means parameter count alone is not a sufficient proxy for adaptation quality.

One plausible interpretation is that cloze-style paraphrase detection relies heavily on representation reshaping in the feed-forward pathway, while joint attention+MLP adaptation adds flexibility by allowing contextual interaction and feature transformation to change together. The scaling plot in Appendix Figure 2 makes the same point graphically: performance is not monotonic in trainable parameter count.

6.3 Moderate Adapter Rank Provides the Good Trade-off

Across the paraphrase experiments, $r = 8$ is consistently the strongest rank within each family. This suggests that once the task-relevant update subspace is accessible, increasing rank mainly adds optimization burden rather than useful capacity. The sonnet results, though noisier, point in the same direction: the best results do not come from the largest adapters or the broadest placement.

6.4 Optimization can matter as much as architecture

One of the clearest lessons of the project is that optimization should be treated as a first-order target. In paraphrase detection, sweeping only the learning rate within a fixed architecture moves

development accuracy from 0.870 to 0.894. In sonnet generation, optimization-related choices also have pronounced effects, though with a less predictable landscape.

This matters for PEFT evaluation more broadly. Reporting a single adapter structure with a baseline learning rate can understate what the method can actually achieve. Architecture determines the shape of the attainable frontier, but optimization determines where a run lands on that frontier. This effect is also visible in Appendix Figure 3, where holding the adapter structure fixed while varying only the learning rate leads to noticeably different optimization trajectories and final accuracies.

6.5 Task asymmetry is the key qualitative finding

The most important high-level conclusion is that LoRA behaves differently across task types. On paraphrase detection, it is robust, efficient, and consistently close to full fine-tuning. On sonnet generation, it can match the baseline, but only in a narrower configuration regime.

A natural interpretation is that low-rank residual adaptation is especially well suited to tasks where downstream adaptation mainly reorients a pretrained representation toward a constrained decision boundary. Open-ended generation is less forgiving because it requires the model to maintain a broad next-token policy over long contexts while also preserving stylistic and local fluency properties. Under that demand, LoRA remains viable but becomes more sensitive to exactly how and where the adaptation is applied.

6.6 Attention-focused LoRA performs best for sonnet generation

The results show that applying LoRA to the attention layers achieves the best performance among the parameter-efficient configurations. In the default attention-only LoRA model with rank $r = 8$, the model reaches a development score of approximately 40.7, which is close to the 41.18 score achieved by full fine-tuning while training far fewer parameters.

In contrast, applying LoRA to the MLP layers or to all linear layers yields lower development scores, suggesting that adapting the attention mechanism is more effective for controlling generation behavior in this task. Since applying Lora does not reduce much of the training time and resources, full fine-tune is suggested to use for small dataset tasks.

7 Conclusion

We studied Low-Rank Adaptation as an efficient alternative to full fine-tuning for GPT-2 on the Stanford CS224N default project tasks. Using a from-scratch GPT-2 implementation initialized with pretrained weights, we varied adapter placement, rank, and learning rate across paraphrase detection and sonnet generation.

Five conclusions emerge. First, LoRA is highly effective for cloze-style paraphrase detection: the best completed configuration reaches 0.894 development accuracy, only 0.005 below full fine-tuning, while training just 1.05% of parameters. Second, adapter placement is decisive: joint attention+MLP adaptation consistently outperforms adapting either pathway alone. Third, moderate rank performs best; increasing rank beyond 8 does not improve results. Fourth, optimization is critical: learning-rate tuning closes most of the remaining paraphrase gap and greatly changes the performance. Fifth, task type matters: LoRA is extremely strong on the structured semantic task and can also match the sonnet baseline, but generation quality is much more configuration-sensitive.

Our conclusions should be interpreted within the scope of the project. The generation dataset for sonnet task is small, chrF is only a partial measure of poetic quality, and all main comparisons are reported on development results within the available compute budget. Even with these limits, the evidence supports a clear practical takeaway: for decoder-only language models, LoRA offers a compelling efficiency–performance trade-off, especially on structured understanding tasks, and can recover nearly all of the benefits of full fine-tuning at a tiny fraction of the adaptation cost.

Team contributions (Required for multi-person team)

Yi Lu was primarily responsible for the LoRA study on paraphrase detection, including experiment design, hyperparameter comparison, adapter-placement analysis, and writing the paraphrase-related sections of the report. Shengzhe (Andy) Zhang was primarily responsible for the sonnet generation portion of the project, including experiment execution, result tracking, and writing the sonnet-related sections. Both authors contributed to the shared base GPT-2 implementation, discussed the overall research framing together, and jointly revised the final report.

References

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.
- [3] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP*, 2018.
- [4] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.
- [5] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.
- [6] Stanford CS224N. CS224N Default Final Project: Build GPT-2. Course handout, 2026.
- [7] Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, 2015.

Appendix

Additional Paraphrase Figures

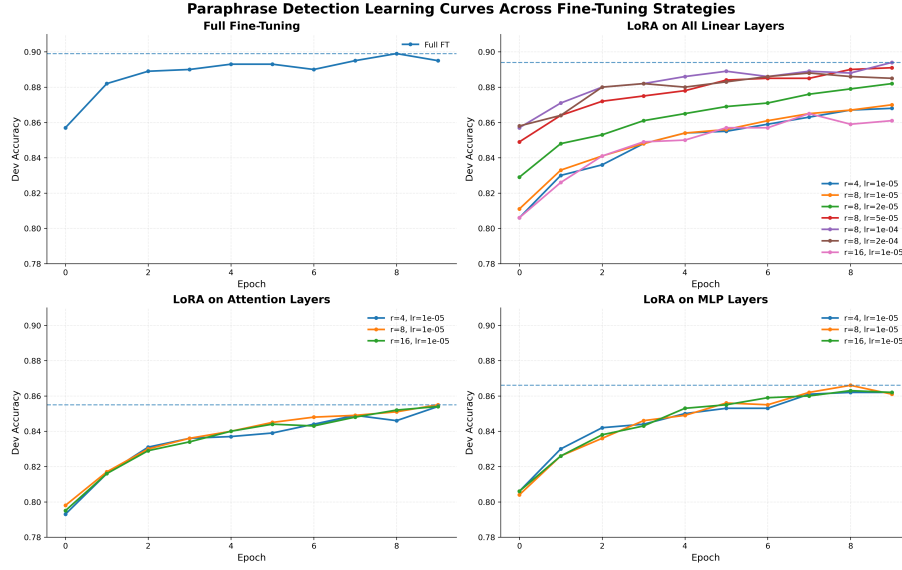


Figure 1: Development accuracy learning curves for paraphrase detection across full fine-tuning and representative LoRA families. The curves show that full fine-tuning reaches the highest final performance, but all-target LoRA with rank $r = 8$ and tuned learning rate converges to a nearby plateau while using far fewer trainable parameters.

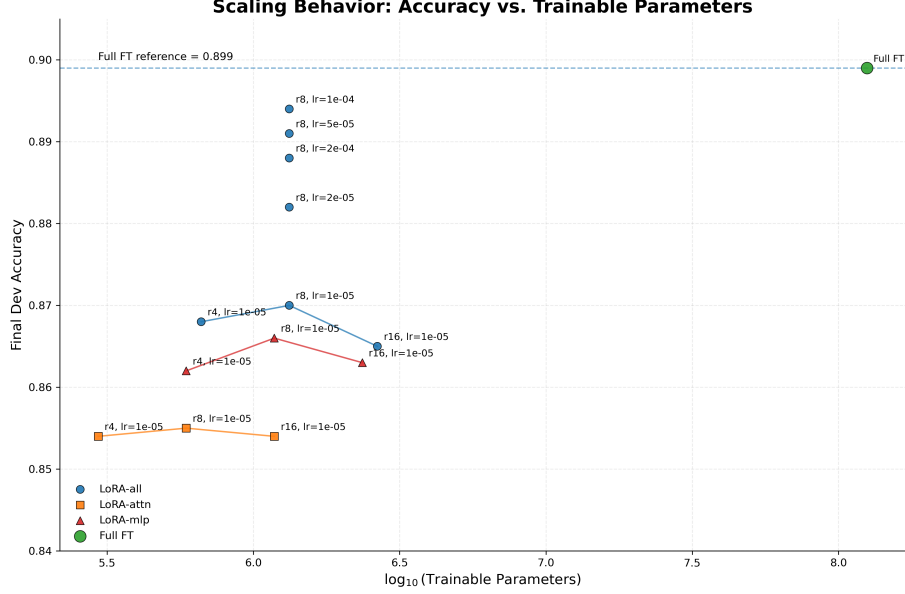


Figure 2: Scaling behavior of paraphrase detection performance as a function of trainable parameters. Increasing adapter size does not monotonically improve accuracy, and adapter placement is more predictive of final performance than parameter count alone. Full fine-tuning is shown as the high-capacity reference point.

Training Dynamics of LoRA (target=all, rank=8) Across Learning Rates

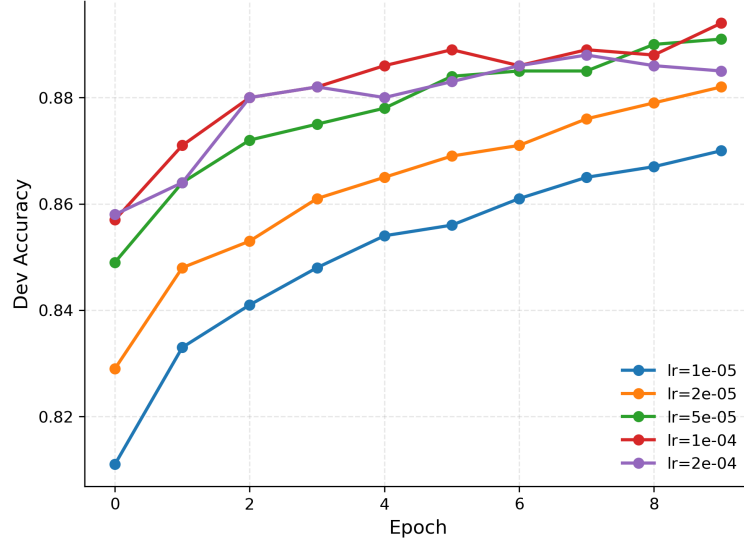


Figure 3: Training dynamics for LoRA with target=all and rank $r = 8$ under different learning rates. Holding architecture fixed while varying only the learning rate reveals that optimization quality strongly affects convergence and final development accuracy.

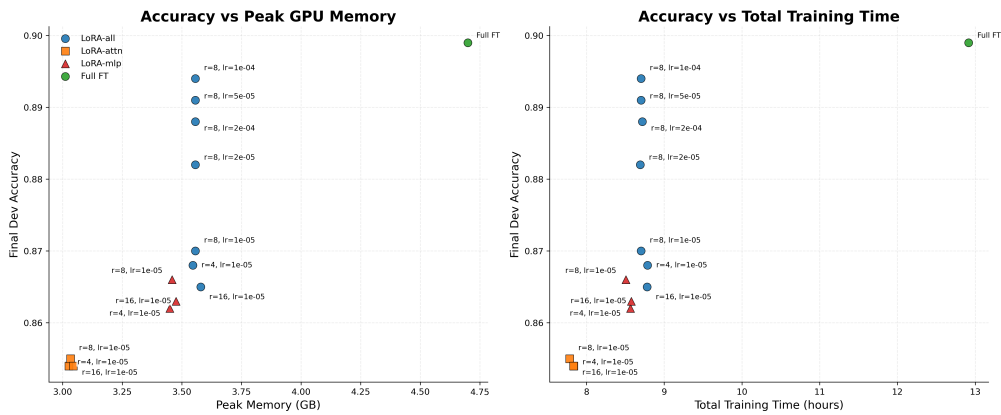


Figure 4: Efficiency–performance trade-offs for paraphrase detection. Left: final development accuracy versus peak GPU memory. Right: final development accuracy versus total training time. Several LoRA configurations approach full fine-tuning performance at substantially lower computational cost.

Additional Sonnet Figures

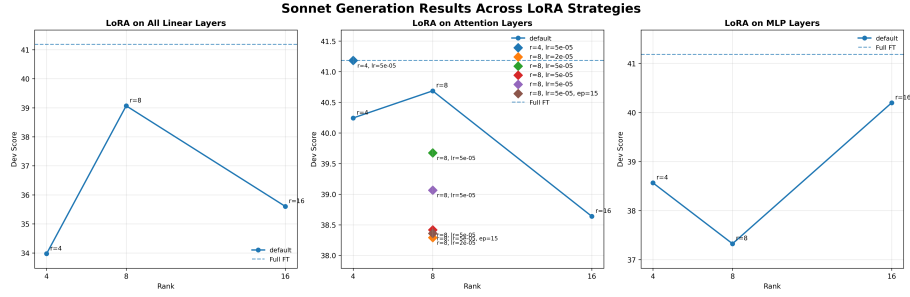


Figure 5: Sonnet-generation comparisons across fine-tuning strategies. The plots compare baseline generation, attention-only LoRA, mlp-only LoRA, and attention+mlp LoRA across rank settings. In contrast to paraphrase detection, generation quality is more sensitive to adapter placement, and broader adaptation does not consistently improve dev chrF.

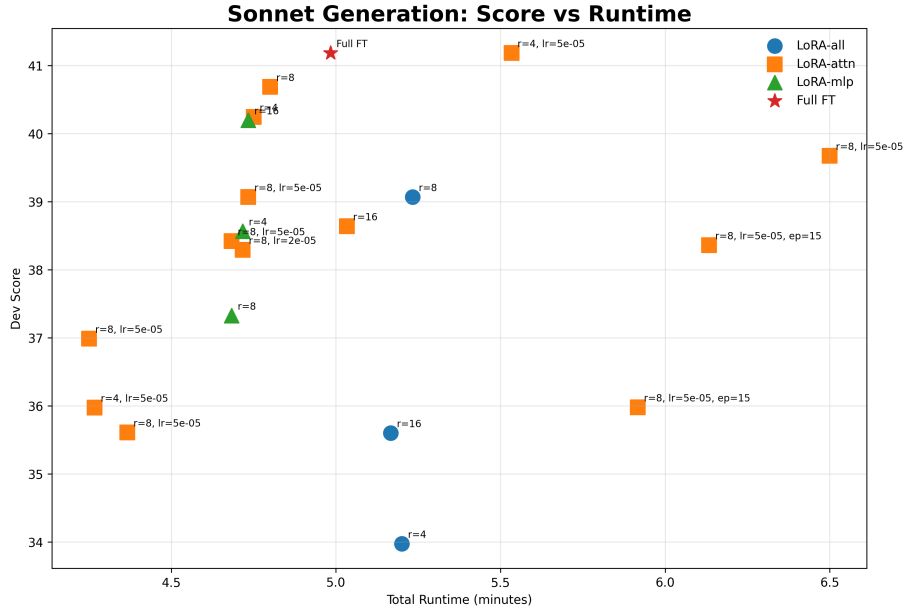


Figure 6: Trade-off between development score and total runtime for sonnet generation. While the baseline remains strong, several LoRA variants achieve competitive chrF at lower or comparable runtime, illustrating the practical efficiency advantages of parameter-efficient fine-tuning even when generation quality is more optimization-sensitive.

Appendix Tables

Family	Setting	Dev Acc.	Dev F1
Full FT	lr = 1e-5	0.899	0.892
Attention-only	r = 4, lr = 1e-5	0.854	0.845
Attention-only	r = 8, lr = 1e-5	0.855	0.847
Attention-only	r = 16, lr = 1e-5	0.854	0.845
MLP-only	r = 4, lr = 1e-5	0.862	0.856
MLP-only	r = 8, lr = 1e-5	0.866	0.859
MLP-only	r = 16, lr = 1e-5	0.863	0.855
Attention+MLP	r = 4, lr = 1e-5	0.868	0.860
Attention+MLP	r = 8, lr = 1e-5	0.870	0.863
Attention+MLP	r = 16, lr = 1e-5	0.865	0.855
Attention+MLP	r = 8, lr = 2e-5	0.882	0.874
Attention+MLP	r = 8, lr = 5e-5	0.891	0.883
Attention+MLP	r = 8, lr = 1e-4	0.894	0.887
Attention+MLP	r = 8, lr = 2e-4	0.888	0.881

Table 3: Complete paraphrase detection performance table.

Family	Setting	Trainable Params	% Trainable	Peak Mem (GB)	Time / Epoch (min)
Full FT	lr = 1e-5	125,030,400	100.000%	4.699	77.50
Attention-only	r = 4, lr = 1e-5	294,912	0.235%	3.026	47.02
Attention-only	r = 8, lr = 1e-5	589,824	0.470%	3.033	46.70
Attention-only	r = 16, lr = 1e-5	1,179,648	0.935%	3.045	47.00
MLP-only	r = 4, lr = 1e-5	589,824	0.470%	3.450	51.40
MLP-only	r = 8, lr = 1e-5	1,179,648	0.935%	3.459	51.05
MLP-only	r = 16, lr = 1e-5	2,359,296	1.852%	3.476	51.45
Attention+MLP	r = 4, lr = 1e-5	663,552	0.528%	3.546	52.71
Attention+MLP	r = 8, lr = 1e-5	1,327,104	1.050%	3.557	52.21
Attention+MLP	r = 16, lr = 1e-5	2,654,208	2.079%	3.579	52.68
Attention+MLP	r = 8, lr = 2e-5	1,327,104	1.050%	3.557	52.15
Attention+MLP	r = 8, lr = 5e-5	1,327,104	1.050%	3.557	52.22
Attention+MLP	r = 8, lr = 1e-4	1,327,104	1.050%	3.557	52.21
Attention+MLP	r = 8, lr = 2e-4	1,327,104	1.050%	3.557	52.32

Table 4: Paraphrase detection efficiency statistics across full fine-tuning and LoRA configurations.

Completed paraphrase experiment matrix

The completed paraphrase experiment suite includes:

- one full fine-tuning baseline,
- attention-only LoRA with $r \in \{4, 8, 16\}$ at learning rate 1×10^{-5} ,
- MLP-only LoRA with $r \in \{4, 8, 16\}$ at learning rate 1×10^{-5} ,
- joint attention+MLP LoRA with $r \in \{4, 8, 16\}$ at learning rate 1×10^{-5} ,
- a learning-rate sweep for joint attention+MLP LoRA at $r = 8$ with

$$\{1 \times 10^{-5}, 2 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}\}.$$

Family	Setting	Dev chrF	Elapsed Time (s)
Baseline	baseline	41.182	299
MLP-only	$r = 4$	38.565	283
MLP-only	$r = 8$	37.324	281
MLP-only	$r = 16$	40.197	284
Attention+MLP	$r = 4$	33.973	312
Attention+MLP	$r = 8$	39.068	314
Attention+MLP	$r = 16$	35.601	310
Attention-only	$r = 4$	40.244	285
Attention-only	$r = 8$	40.687	288
Attention-only	$r = 16$	38.637	302
Attention-only	$r = 4, \text{lr}=5 \times 10^{-5}$	41.182	332
Attention-only	$r = 4, \text{lr}=5 \times 10^{-5}, \text{dropout} = 0.05$	35.972	256
Attention-only	$r = 8, \text{lr}=2 \times 10^{-5}$	38.290	283
Attention-only	$r = 8, \text{lr}=5 \times 10^{-5}$	36.987	255
Attention-only	$r = 8, \text{lr}=5 \times 10^{-5}, \text{dropout} = 0.05$	39.066	284
Attention-only	$r = 8, \text{lr}=5 \times 10^{-5}, \alpha = 32$	38.417	281
Attention-only	$r = 8, \text{lr}=5 \times 10^{-5}, 15 \text{ epochs}$	35.975	355
Attention-only	$r = 8, \text{lr}=5 \times 10^{-5}, \text{temp} = 0.8$	35.608	262
Attention-only	$r = 8, \text{lr}=5 \times 10^{-5}, 15 \text{ epochs}, \text{temp} = 0.8$	38.361	368

Note: Unless otherwise specified in the **Setting** column, experiments use the default hyperparameters from the baseline configuration: learning rate 1×10^{-5} , LoRA rank $r = 8$, $\alpha = 16$, dropout = 0.0, batch size = 8, training epochs = 10, temperature = 1.2, and top- $p = 0.9$.

Table 5: Complete sonnet generation development results from the completed experiment sheet. Results are grouped by adapter family and hyperparameter setting for consistency with the main-text analysis.

Completed sonnet experiment matrix

The completed sonnet experiment sheet includes:

- baseline generation,
- attention-only, mlp-only, and attention+mlp rank sweeps,
- additional attention-focused variants involving learning rate, dropout, alpha, longer training, and decoding temperature.

Practical takeaways

From an engineering standpoint, the project suggests three practical takeaways:

- On structured semantic tasks, LoRA can be a near-drop-in replacement for full fine-tuning with dramatically lower adaptation cost.
- Adapter placement and learning rate matter more than simply increasing rank.
- Open-ended generation remains feasible under LoRA, but the margin for misconfiguration is much smaller and need further tuning.