

GPT-2 with LoRA Optimization: Final Report

Stanford CS224N Default Project

Baihan Zhang
Department of Computer Science
Stanford University
baihan@stanford.edu

Janhavi Purkar
Department of Computer Science
Stanford University
janhavip@stanford.edu

Illia Shkirko
Department of Computer Science
Stanford University
illia@stanford.edu

Abstract

Large pretrained transformers are typically adapted via full fine-tuning, requiring storing task-specific copies of all parameters. We evaluate whether Low-Rank Adaptation (LoRA) [1] provides a parameter-efficient alternative for adapting GPT-2 to sentiment classification, paraphrase detection, and sonnet generation. We implement LoRA adapters in GPT-2 attention and compare three fine-tuning regimes: last-layer tuning, full fine-tuning, and LoRA-based optimization. We also investigate the effect of weight decay on optimizer stability and downstream task performance. Results show that LoRA achieves competitive performance while updating only a small fraction of model parameters, matching or approaching full fine-tuning on classification tasks while underperforming on generation. These findings highlight both the strengths and limitations of parameter-efficient fine-tuning when applied across diverse NLP tasks.

1 Key Information to include

- Mentor: Wei Liu
- External Collaborators (if you have any): N/A
- Sharing project: N/A
- Late Days:

Team member	Late days remaining
Baihan Zhang	5
Janhavi Purkar	5
Illia Shkirko	4
Total pooled	14
Team size	3
Late days applied for the report	3

2 Introduction

Large pretrained transformers have become the dominant approach for natural language processing, achieving state-of-the-art performance across a wide range of tasks. However, adapting these models to downstream tasks typically requires full fine-tuning, where all model parameters are updated for each new task. For modern language models containing hundreds of millions or billions of parameters,

this approach is computationally expensive. These costs make full fine-tuning inefficient in settings where models must be adapted to many tasks or deployed in resource-constrained environments.

Parameter-efficient fine-tuning (PEFT) methods aim to address this challenge by adapting pretrained models while updating only a small subset of parameters. Instead of modifying all weights, these methods introduce lightweight task-specific components while keeping the majority of the pretrained model frozen. Recent approaches such as adapters, prompt tuning [2], prefix tuning [3], and BitFit [4] demonstrate that competitive performance can often be achieved while training only a small fraction of the parameters.

In this work, we ask the following scientific question: can LoRA-based post-training [1] provide a parameter-efficient alternative to full fine-tuning of a pretrained GPT-2 model when applied to diverse downstream tasks such as sentiment analysis, paraphrase detection, and sonnet generation? We investigate whether low-rank adaptation provides a parameter-efficient alternative to full fine-tuning while maintaining competitive task performance.

We choose this question because LoRA was originally proposed and evaluated primarily on very large language models. It remains important to understand whether the same low-rank adaptation principle remains effective for smaller models such as GPT-2. Additionally, evaluating LoRA across multiple task domains—including classification, semantic comparison, and text generation—allows us to examine whether LoRA provides consistent benefits across different types of NLP problems.

To study this question, we implement LoRA within the attention layers of GPT-2 and evaluate its effectiveness on the CS224N default project tasks. We compare three training regimes: training only the final classification layer, full model fine-tuning, and LoRA-based adaptation. Our experiments show that LoRA achieves performance comparable to full fine-tuning on classification tasks while updating only a small fraction of model parameters, though it performs less competitively on the generation task. These findings highlight both the strengths and limitations of parameter-efficient fine-tuning methods when applied across different NLP settings.

3 Related Work

Hu et al. [1] introduce LoRA (Low-Rank Adaptation), a parameter-efficient fine-tuning method that represents task-specific weight updates as low-rank matrices while keeping all pretrained weights frozen, drastically reducing the number of trainable parameters. These low-rank updates are merged into the original weights at deployment which does not affect inference speed or model structure. This technique is shown to greatly reduce optimizer memory during training and task-specific checkpoint sizes. We use this paper as a primary source of guidance for our project.

We surveyed a few related papers that introduce PEFT (parameter-efficient fine-tuning) ideas similar to LoRA.

Lester et al. [2] propose prompt tuning, which learns a small set of continuous prompt embeddings prepended to the input sequence while keeping all model parameters frozen. They show that prompt tuning achieves competitive performance with full fine-tuning on large models while training only a small number of parameters. Unlike LoRA, which introduces low-rank updates inside transformer layers, prompt tuning modifies only the input representation.

Ben-Zaken et al. [4] introduce BitFit, a parameter-efficient fine-tuning method that updates only the bias parameters of a pretrained transformer while freezing all other weights. Despite training only a small fraction of parameters, BitFit achieves competitive results on several NLP tasks. In contrast to LoRA, which adds trainable low-rank matrices to attention weights, BitFit relies solely on adjusting existing bias parameters.

Houlsby et al. [5] propose adapter tuning, which inserts small bottleneck modules after each sublayer of a pretrained transformer and trains only those while keeping all original weights frozen. They show that adapters achieve near full fine-tuning performance on GLUE while adding only 3% task-specific parameters. Unlike LoRA, adapter layers introduce additional inference latency since they add depth to the network.

Li and Liang [3] propose prefix-tuning, which prepends a sequence of trainable continuous vectors to each transformer layer while keeping all model parameters frozen. Applied to GPT-2 for table-to-text generation, prefix-tuning matches full fine-tuning performance using only 0.1% of the parameters. In

contrast to LoRA, prefix-tuning reduces the usable context length and was found to be less stable to optimize.

Together, these methods demonstrate that large pretrained language models can be adapted efficiently by updating only a small subset of parameters, motivating our evaluation of LoRA as a parameter-efficient alternative to full fine-tuning.

4 Approach

We investigate whether Low-Rank Adaptation (LoRA) [1] can provide a parameter-efficient alternative to full fine-tuning of GPT-2 on the CS224N default tasks. We compare three regimes:

- **Last-linear-layer:** GPT-2 frozen, only task head trained,
- **Full-model:** all GPT-2 parameters updated,
- **LoRA:** selected attention projections replaced with low-rank adapters.

4.1 Baseline: GPT-2 for Classification and Generation

We build directly on the provided CS224N GPT-2 implementation. For both sentiment classification (SST, CFIMDB) and paraphrase detection (Quora), we use the final token embedding as the sequence representation. For sentiment classification, we apply a linear layer to the final token embedding to produce logits over labels, trained with cross-entropy loss. The fine-tuning mode determines whether GPT-2 parameters are frozen (last-linear-layer) or updated (full-model).

For paraphrase detection, we apply a two-dimensional linear head to the final token embedding and compute cross-entropy over the GPT-2 token indices corresponding to “yes” and “no.” For sonnet generation, we fine-tune the model autoregressively on Shakespeare sonnets, where given the first three lines as a prompt, the model generates the remaining lines token by token, trained with cross-entropy loss. Optimizer uses the provided AdamW implementation.

Baseline Extension Note that we run preliminary experiments to determine optimal hyperparameters. While more details are shared in the upcoming sections of the report, it’s important to note that in all experiments, we implemented gradient clipping, weight decay, and linear warmup scheduling in addition to the provided baseline.

4.2 LoRA Integration

Our primary extension is a custom `LoRALinear` module that augments a linear layer with a low-rank update while freezing the base weights. For input x : $y = Wx + \frac{\alpha}{r}BAx$, where W is frozen, and $A \in \mathbb{R}^{r \times d_{in}}$, $B \in \mathbb{R}^{d_{out} \times r}$ are trainable parameters. The rank r controls the dimensionality of the low-rank update and therefore the number of additional trainable parameters introduced by LoRA.

When LoRA is enabled, selected attention projection layers (query, key, value) in GPT-2 are replaced with `LoRALinear`. The base weights inside these layers are frozen by construction. The task-specific classification head remains fully trainable. Freezing of the remaining GPT-2 parameters is controlled separately by the fine-tuning mode, allowing direct comparison between: Full-model fine-tuning, Last-linear-layer fine-tuning, and LoRA-based adaptation with low-rank updates in attention layers.

We have coded the implementation ourselves based on the original LoRA paper [1].

5 Experiments

We ran multiple preliminary experiments to achieve the highest performance of the baseline GPT-2 model on sentiment analysis, paraphrase detection, and sonnet generation tasks. We then run experiments on GPT-2 with LoRA with more details described below.

5.1 Data

We evaluate our implementation on the CS224N default project datasets: SST [6], CFIMDB [7], Quora Question Pairs (paraphrase detection), and Shakespeare Sonnets (generation). SST consists of 8,544 training and 1,101 dev examples of single sentences from movie reviews with five sentiment labels. CFIMDB consists of 1,701 training and 245 dev examples of longer movie reviews with binary sentiment labels. The Quora dataset contains question pairs with binary paraphrase labels. The sonnet dataset consists of 143 training sonnets and 12 held-out test sonnets written by Shakespeare. All datasets use the fixed train/dev splits provided in the course handout. Inputs are tokenized using GPT-2 BPE via the provided tokenizer utilities. No additional preprocessing beyond the default setup was applied.

5.2 Evaluation method

For SST, CFIMDB, and Quora we use development set accuracy. For sonnet generation, we use the chrF metric, which is a character-level n -gram F-score computed against Shakespeare’s original held-out sonnets using the provided `evaluation.py` pipeline.

5.3 Experimental details

We ran three sets of experiments:

- Preliminary tuning meant to find the best hyperparameters for our baselines.
- LoRA integration to assess the impact of introducing this optimization.
- LoRA hyperparameter search where we altered LoRA-specific hyperparameters.

5.3.1 Preliminary experiments

To begin with, we evaluated the effectiveness of various hyperparameters. We specifically focused on gradient clipping, warmup scheduling, and weight decay to optimize the performance of the baseline implementation prior to applying LoRA. These three modifications were applied as fixed settings across all experiments.

We applied gradient clipping with `max_norm = 1.0` to all experiments. This prevents exploding gradients during fine-tuning, which can destabilize training particularly in the early epochs when parameter updates are large. We used a linear warmup schedule over the first 10% of total training steps, followed by linear decay to 0. This allows the model to stabilize at the start of training before the learning rate reaches its peak value, which is especially important when fine-tuning from pretrained weights. Finally, we introduced weight decay of 0.01 to the AdamW optimizer, applied only to weight matrices while excluding biases and LayerNorm parameters. Weight decay acts as a form of regularization by penalizing large parameter values, which can help improve generalization and reduce overfitting.

5.3.2 GPT-2 with LoRA

We run multiple experiments to assess the effectiveness of LoRA optimization. In particular, we computed baseline and LoRA accuracies on sentiment analysis, paraphrase detection, and sonnet generation tasks. LoRA learning rate was set to 1×10^{-4} , 3×10^{-4} , and 5×10^{-4} , which is about $10 \times$ higher than full fine-tuning (4×10^{-5}), following standard LoRA practice since low-rank matrices are initialized near zero. Other hyperparameters remained the same as in preliminary experiments.

As for the training times, each epoch of the paraphrase detection task took about 20 minutes when run on an A100 GPU.

5.3.3 LoRA hyperparameters

LoRA was applied to the query, key, and value projection matrices in each of the 12 transformer layers, with rank $r = 8$ and scaling factor $\alpha = 16$ as the default configuration. We then ran an ablation over $\alpha \in \{8, 16, 32\}$ at `lr=1e-4` to assess the sensitivity of results to this hyperparameter. On SST and CFIMDB, α had no meaningful impact — the best dev accuracy was identical across all

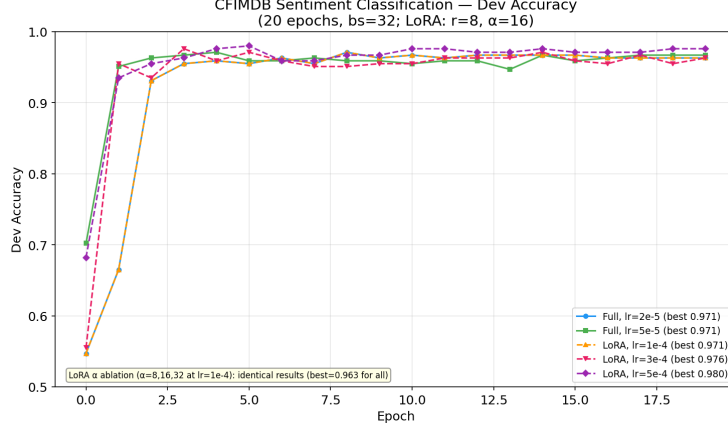


Figure 1: Dev accuracy on CFIMDB sentiment classification across epochs for full fine-tuning and LoRA with different learning rates

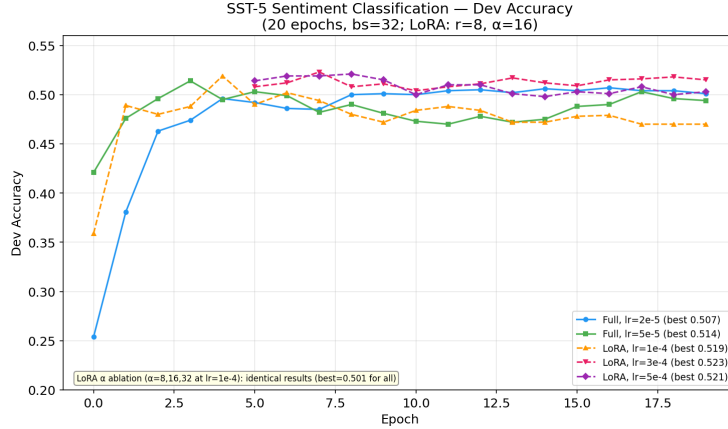


Figure 2: Dev accuracy on SST-5 sentiment classification across epochs for full fine-tuning and LoRA variants.

three values. On sonnet generation, α had a more meaningful effect: chrF scores were 37.1, 36.9, and 40.2 for $\alpha = 8, 16, 32$ respectively. The paraphrase detection α ablation was not completed due to training preemption.

5.4 Results

While we ran many additional evaluations, in this section, we present the most relevant and striking results. We will primarily discuss the impact of introducing LoRA to attention layers. We will also talk about how we tuned LoRA’s hyperparameters to achieve the best empirical results.

Sentiment Analysis Results on both sentiment tasks are shown in Figures 1 and 2. On CFIMDB, full fine-tuning at $lr=2e-5$ and $lr=5e-5$ both converged quickly and reached a best dev accuracy of 0.971. LoRA runs at higher learning rates showed a notable dip at epoch 0 before recovering rapidly, with $lr=3e-4$ reaching 0.976 and $lr=5e-4$ reaching 0.980. The α ablation at $lr=1e-4$ had no meaningful impact, with best accuracy of 0.963 across all three values. On SST-5, all runs showed more variance across epochs. Full fine-tuning achieved best accuracies of 0.507 ($lr=2e-5$) and 0.514 ($lr=5e-5$). LoRA at $lr=3e-4$ achieved the best result of 0.523, outperforming both full fine-tuning runs. As with CFIMDB, the α ablation had no effect on SST, with best accuracy of 0.501 across all values at $lr=1e-4$.

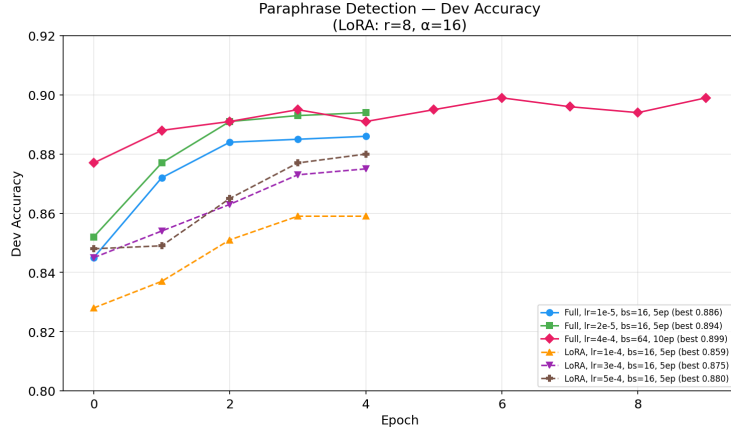


Figure 3: Dev accuracy on paraphrase detection across epochs comparing full fine-tuning and LoRA models.

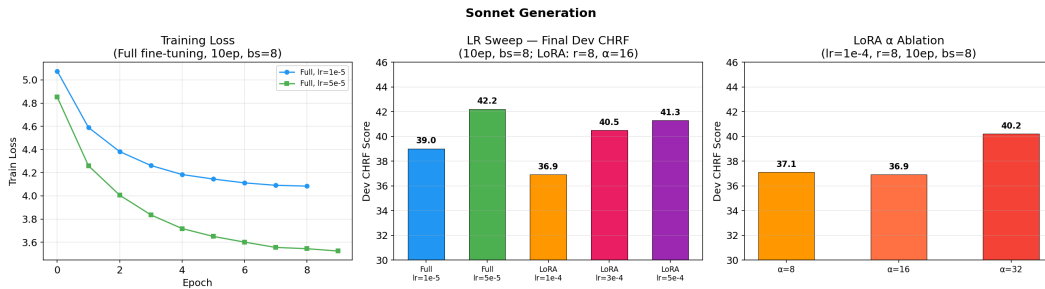


Figure 4: Sonnet generation results: training loss curves for full fine-tuning vs LoRA and chrF evaluation scores for different learning rates and LoRA hyperparameters.

Paraphrase Detection On the test Gradescope submission, we obtained a result of **0.85**. The rest of the results on paraphrase detection are shown in Figure 3. The best full fine-tuning result was 0.899, achieved with $lr=4e-4$, $bs=64$ over 10 epochs. Runs at lower learning rates ($lr=1e-5$ and $lr=2e-5$) with $bs=16$ over 5 epochs reached 0.886 and 0.894 respectively. All LoRA runs converged more slowly and to lower final accuracies, with best results of 0.859, 0.875, and 0.880 for $lr=1e-4$, $3e-4$, and $5e-4$ respectively. LoRA did not close the gap with full fine-tuning on this task.

Sonnet Generation On the test Gradescope submission, we obtained a result of **41.98**. The rest of the results on sonnet generation are shown in Figure 4. Training loss curves for both full fine-tuning runs decrease steadily over 10 epochs, with $lr=5e-5$ reaching a lower final loss than $lr=1e-5$. In the LR sweep, full fine-tuning at $lr=5e-5$ achieved the best chrF score of 42.2, while LoRA at $lr=5e-4$ achieved 41.3 and $lr=3e-4$ achieved 40.5. LoRA at $lr=1e-4$ scored only 36.9, below the untuned baseline of 39.0. The α ablation showed meaningful sensitivity, with chrF scores of 37.1, 36.9, and 40.2 for $\alpha = 8, 16, 32$ respectively.

Summary Table 1 and Figure 5 summarize results across all tasks. LoRA matched or exceeded full fine-tuning on both sentiment tasks, with a best SST-5 accuracy of 0.523 vs 0.514 and a best CFIMDB accuracy of 0.980 vs 0.971. On paraphrase detection and sonnet generation, full fine-tuning retained an advantage, with best accuracies of 0.899 vs 0.880 and chrF scores of 42.2 vs 41.3 respectively. All tuned models improved over the untuned baseline across every task.

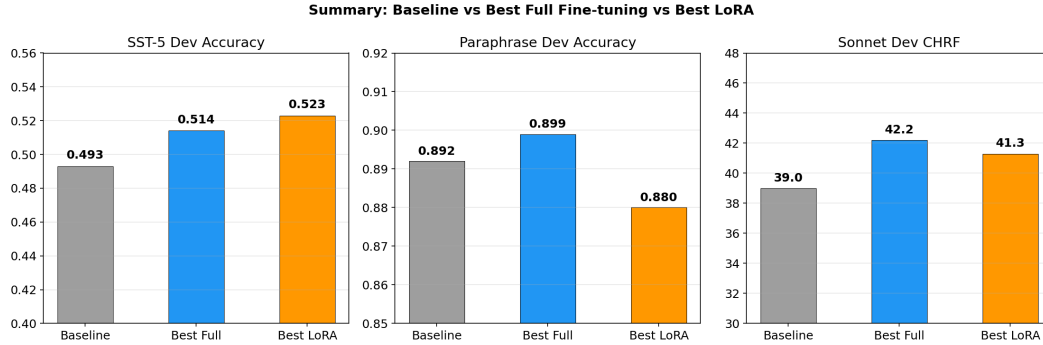


Figure 5: Performance comparison between full fine-tuning, last-layer tuning, and LoRA on all workloads

Task	Metric	Batch	Epochs	LR	LoRA LR	Last-Linear	Full FT	LoRA
SST	Dev Acc	8	10	10^{-5}	10^{-4}	0.453	0.514	0.523
CFIMDB	Dev Acc	8	10	10^{-5}	10^{-4}	0.878	0.971	0.980
Paraphrase	Dev Acc	16	5	$4 \cdot 10^{-5}$	$4 \cdot 10^{-4}$	0.720	0.899	0.880
Sonnet	chrF	8	10	$5 \cdot 10^{-5}$	$5 \cdot 10^{-4}$	39.0	42.2	41.3

Table 1: Training hyperparameters and development results across tasks.

6 Analysis

6.1 Why LoRA works better on some tasks than others

By analyzing the results from Figures 1–5, we see that LoRA performs much better on classification tasks (SST, CFIMDB). Classification tasks require relatively small modifications to the pretrained representation space, so small low-rank updates are sufficient. In contrast, paraphrase detection requires a more complex semantic comparison, which may require a higher number of trainable weights. Finally, generation tasks (e.g. sonnet generation) require modeling even more complex distributions, so full fine-tuning works best. We conclude that LoRA shows a more significant performance drop in more complex tasks.

6.2 Parameter-efficiency vs Performance tradeoff

A key motivation behind LoRA is reducing the number of trainable parameters while maintaining competitive performance. In our experiments, full fine-tuning updates all GPT-2 parameters, whereas LoRA updates only the low-rank matrices A and B . This significantly reduces the number of trainable parameters while preserving most of the model capacity through the frozen pretrained weights. **Trainable Parameter Count** per layer is $3 \times (768 \times 8 + 8 \times 768) = 36,864$, and across 12 layers, it’s 442,368 trainable parameters, which means that LoRA updates represent approximately 0.36% of GPT-2’s 124M parameters. Despite updating far fewer parameters, LoRA achieved performance comparable to full fine-tuning on both sentiment tasks and remained competitive on generation. This suggests that the majority of the useful knowledge required for these tasks is already present in the pretrained model, and only small task-specific adjustments are necessary.

6.3 α hyperparameter sensitivity

Sonnet generation shows clear sensitivity to the LoRA scaling parameter α , unlike classification tasks where α had no effect. Based on Figure 4, we see that increasing α to 32 has significantly improved the performance. Naturally, generation tasks require more expressive adaptation, so a higher α improves performance.

6.4 Training dynamics

Based on the Figures 1–4, we noticed that LoRA’s runs initially suffer performance loss before recovering. As a result, LoRA converges more slowly when compared with full-finetuning. The reason behind the initial dip is the 0-initialization of some of the LoRA matrices, meaning the effective update to the pretrained weight matrix is initially very small. Early training steps may produce unstable gradients while the adapters begin learning useful updates. Despite slower early learning, LoRA eventually approaches or matches full-finetuning on classification tasks.

6.5 LoRA sensitivity to learning rate

Some LoRA matrices are initialized near zero, so larger learning rates are needed. We have found that a $10\times$ increase compared to the optimal full fine-tuning learning rate typically provides the best results. In our plots, we noticed no clear pattern for further alteration of learning rates. This suggests that empirical hyperparameter fine-tuning is necessary to achieve the best performance with PEFT methods such as LoRA.

6.6 Dataset size considerations

Dataset size varies a lot across our benchmarks. SST is the largest dataset with 8,544 training examples, while CFIMDB has only 1,701 and the sonnet dataset is much smaller still, hence, generation may largely rely on pretraining of GPT-2. On smaller datasets, full fine-tuning may risk overfitting because a large number of parameters are updated. Given our results, it is fair to suggest that LoRA might help prevent overfitting on small datasets.

7 Conclusion

In this work, we evaluated whether LoRA provides a parameter-efficient alternative to full fine-tuning of GPT-2 across sentiment classification, paraphrase detection, and sonnet generation. We emphasize that the goal is not to outperform the baseline, but rather to match or get close enough to prove that PEFT achieves comparable results. We implemented LoRA adapters across the query, key, and value matrices of GPT-2, and compared performance against an untuned baseline and best full fine-tuning.

On SST-5, LoRA (0.523) outperformed both the baseline (0.493) and best full fine-tuning (0.514), suggesting that low-rank adaptation is sufficient and even slightly beneficial for sentiment classification. On CFIMDB, LoRA (0.980) exceeded full fine-tuning (0.971), further supporting this trend. Paraphrase detection told a different story: LoRA (0.880) underperformed both full fine-tuning (0.899) and the untuned baseline (0.892), indicating that the task may require broader parameter updates than low-rank adapters allow. For sonnet generation, LoRA (chrF = 41.3) fell short of full fine-tuning (42.2) but still improved over the baseline (39.0).

Our α ablation showed that classification tasks were insensitive to this hyperparameter, while sonnet generation was more sensitive, with chrF scores of 37.1, 36.9, and 40.2 for $\alpha = 8, 16, 32$ respectively. Overall, LoRA is an effective strategy for classification tasks but falls short on tasks requiring more expressive updates, such as paraphrase detection and generation. Future work could explore higher-rank adapters or different layer selection strategies to address these limitations.

Team contributions

Illia: Implemented the attention module and GPT-2 transformer layer.

Janhavi: Implemented the GPT-2 model, optimizer, and paraphrase detection.

Baihan: Implemented the sentiment classifier and sonnet generation.

All other work, including LoRA development, hyperparameter tuning, experiments, and report writing, was split equally between all three team members.

References

- [1] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.
- [2] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.
- [3] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 4582–4597, 2021.
- [4] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.
- [5] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.
- [6] 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 (EMNLP)*, pages 1631–1642, 2013.
- [7] Divyansh Kaushik, Eduard Hovy, and Zachary C. Lipton. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations (ICLR)*, 2020.