

Comparing ReFT and LoRA on Classification and Generative Tasks with GPT-2

Stanford CS224N Default Project

Ryan Catullo

Department of Computer Science
Stanford University
rcatullo@stanford.edu

Abstract

We compare two parameter-efficient fine-tuning methods for GPT-2, LoRA and ReFT, against standard full fine-tuning on sentiment classification, paraphrase detection, and sonnet generation. The main question is whether a small trainable parameter budget can recover most of the performance of full fine-tuning, and whether those parameter savings lead to real runtime gains. Across the classification tasks, LoRA is the strongest parameter-efficient method and comes especially close to full fine-tuning on Quora paraphrase detection, where it reaches 88.1% dev accuracy versus 89.6% for full fine-tuning while training about 1.05% of the parameters. ReFT is more parameter-efficient and often faster in wall-clock time, but it consistently gives up more quality. On sonnet generation, both PEFT methods lag far behind full fine-tuning, which suggests that this small adaptation budget is much more limiting for open-ended generation than for classification.

1 Key information

- Team members: Ryan Catullo (rcatullo@stanford.edu)
- Custom or Default Project: Default
- Mentor: None
- Sharing project: No
- External collaborators: None

2 Introduction

Parameter-efficient fine-tuning has become an attractive alternative to full fine-tuning because it can adapt a pretrained language model while updating only a small fraction of its parameters. This is especially appealing for decoder-only models such as GPT-2, where full fine-tuning quickly becomes expensive in memory, storage, and checkpoint size as model size grows. At the same time, parameter efficiency is only useful if it preserves enough downstream performance and ideally offers practical runtime benefits.

This project studies that tradeoff by comparing full fine-tuning, LoRA, and ReFT on the same GPT-2 small backbone across both classification and generation settings. We focus on three task families from the CS224N default project: sentiment classification, paraphrase detection, and sonnet generation. These tasks differ in structure and difficulty, which makes them useful for testing whether parameter-efficient adaptation behaves differently on short label-prediction problems versus open-ended generation.

Our results show that the answer is yes. LoRA is the strongest PEFT method in our setup and comes close to full fine-tuning on the classification tasks, especially Quora paraphrase detection. ReFT is

even more parameter-efficient and often faster in wall-clock time, but it consistently trails LoRA in quality. On sonnet generation, both PEFT methods lag substantially behind full fine-tuning. This suggests that the same low-rank or representation-level adaptation budget that works reasonably well for label-oriented tasks is much more limiting for sequence generation.

3 Related Work

Parameter-efficient fine-tuning has become a practical alternative to full fine-tuning because it reduces memory use and the number of trainable parameters without requiring a new full model for every task. LoRA is one of the most widely used PEFT methods. It freezes the pretrained model weights and learns low-rank updates to selected layers, which often works well in practice with a very small parameter budget [1]. ReFT takes a different approach and learns interventions directly in representation space rather than editing the original weights [2]. This makes it interesting to compare against LoRA because the two methods are efficient for different reasons.

We use both the Hugging Face PEFT library and the StanfordNLP pyreft library as implementation resources when writing our code. Our work is not proposing a new PEFT method. Instead, it is a controlled empirical comparison of full fine-tuning, LoRA, and ReFT on the same model and across multiple tasks, with attention to both end quality and practical runtime behavior.

4 Approach

We use GPT-2 small [3] as the base model in all experiments. The project compares three adaptation strategies: full fine-tuning, LoRA, and ReFT.

Full fine-tuning updates all model parameters. This is the strongest baseline in terms of adaptation capacity, but it is also the most expensive in trainable parameters.

LoRA freezes a pretrained weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and learns a low-rank update

$$\Delta W = BA$$

where $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}} \times r}$ with $r \ll \min(d_{\text{in}}, d_{\text{out}})$ [1]. The adapted layer computes

$$Wx + \Delta Wx = Wx + B(Ax)$$

so the number of trainable parameters is reduced from $d_{\text{out}}d_{\text{in}}$ to $r(d_{\text{in}} + d_{\text{out}})$.

ReFT freezes the original model weights and instead learns small interventions on hidden representations [2]. In the low-rank form used here, a hidden state $h \in \mathbb{R}^d$ is modified as

$$h' = h + U(V^\top h)$$

where $U, V \in \mathbb{R}^{d \times r}$. Intuitively, LoRA changes the model in weight space, while ReFT changes the model in representation space.

We study three task settings. The first is sentiment classification on SST and CFIMDB. The second is paraphrase detection on Quora question pairs. For Quora, we keep the cloze-style setup which casts the task as conditional generation of a yes or no answer token. The third task is sonnet generation, which provides a generative setting rather than a discrete classification task. This mix of tasks makes it possible to see whether PEFT behaves differently on tasks that require short label decisions versus tasks that require longer-form text generation.

5 Experiments

5.1 Data

We evaluate on four datasets. For sentiment classification, we use the Stanford Sentiment Treebank and the course-provided CFIMDB dataset. SST is a standard fine-grained sentiment benchmark introduced by Socher et al. [4]. CFIMDB is a highly polar movie-review classification dataset distributed as part of the CS224N default project, and it is closely related in domain to prior IMDb sentiment data [5]. For paraphrase detection, we use the Quora question-pair dataset distributed with

the default project. For generation, we use the Shakespearean sonnet corpus provided in the same handout.

These tasks were chosen because they stress different aspects of adaptation. SST and CFIMDB test sentence- and document-level sentiment classification. Quora is still a classification task, but it is framed as next-token prediction over yes and no answers. Sonnet generation is the most open-ended setting and requires the model to adapt its language modeling behavior rather than just sharpen a short decision boundary.

5.2 Evaluation method

For SST, CFIMDB, and Quora, we report best dev accuracy and best dev F1. These are the main metrics used to compare the classification runs. For sonnet generation, we report best training loss, best training perplexity, and final token accuracy, since the logged outputs for this task are language modeling metrics rather than held-out classification scores.

In the plots below, the accuracy curves for SST, CFIMDB, and Quora use dev accuracy. The sonnet accuracy curve uses training token accuracy. The loss curves use training loss for all four datasets.

5.3 Experimental details

All experiments use GPT-2 small with learning rate 10^{-4} . For sentiment classification, LoRA is applied to attention and MLP modules with rank $r = 8$, $\alpha = 16$, and dropout 0.1. For paraphrase detection, LoRA is applied to attention modules with the same rank and scaling. ReFT uses rank $r = 8$, $\alpha = 16$, dropout 0.1, and all-layer interventions. Sentiment runs train for 5 epochs, paraphrase runs for 10 epochs, and sonnet runs for 30 epochs.

All runs were trained on an NVIDIA L40S GPU. One practical goal of the project was to see whether parameter-efficient fine-tuning provides noticeable end-to-end runtime gains in addition to reducing the number of trainable parameters. To make these comparisons meaningful, we did some batch-size exploration for each task to push GPU memory usage high enough that the runs were close to compute-limited rather than bottlenecked by tiny batches. We also used pinned memory and parallel dataloaders to reduce input-pipeline overhead.

We did additional hyperparameter tuning on the smaller sentiment datasets, SST and CFIMDB, using a small grid search over learning rates. This helped refine both PEFT implementations. In particular, for LoRA we added a trainable MLP classification head for the sentiment tasks, and for ReFT we moved from last-layer-only interventions to all-layer interventions. Those changes substantially improved the effective adaptation capacity and downstream performance.

For implementation guidance, the ReFT code was informed by the StanfordNLP pyreft library [6], and the LoRA code was informed by the Hugging Face PEFT library [7].

5.4 Results

Table 1 shows the final classification results. LoRA is the strongest PEFT method on all three classification datasets. It is especially competitive on Quora, where it reaches 88.1% dev accuracy compared to 89.6% for full fine-tuning while training only about 1.05% of the parameters. On CFIMDB, LoRA also stays close to full fine-tuning, while ReFT falls much further behind. SST is the weakest dataset for all three methods, which suggests the task setup itself is harder for GPT-2 small and leaves less headroom for PEFT to match the full model.

Table 2 shows that these results do not generalize to sonnet generation. Full fine-tuning reaches much lower loss and perplexity than either PEFT method, and its final token accuracy is far higher. This suggests that the low-rank and representation-level adaptations used here are much more effective for label-oriented tasks than for learning a stronger generative distribution over long outputs, at least with GPT-2 small and the parameter budgets used here.

Figures 1, 2, and 3 show how accuracy evolves over training. On SST, all three methods improve early, but full fine-tuning peaks after one epoch and then declines slightly while training loss continues to fall later on. That pattern suggests overfitting. On CFIMDB, full fine-tuning saturates very quickly, while LoRA starts slower but steadily closes most of the gap by the final epoch. ReFT improves

Task	Method	Trainable params	Trainable %	Best dev acc	Best dev F1
SST	Full fine-tune	125,034,245	100.0000%	0.5159	0.4968
SST	LoRA	1,330,949	1.0533%	0.4896	0.4119
SST	ReFT	151,301	0.1209%	0.4723	0.3941
CFIMDB	Full fine-tune	125,031,938	100.0000%	0.9755	0.9755
CFIMDB	LoRA	1,328,642	1.0515%	0.9633	0.9632
CFIMDB	ReFT	148,994	0.1190%	0.7265	0.7205
Quora	Full fine-tune	125,031,938	100.0000%	0.8960	0.8886
Quora	LoRA	1,327,104	1.0503%	0.8814	0.8749
Quora	ReFT	147,456	0.1178%	0.8300	0.8177

Table 1: Classification results across tasks. LoRA is the strongest PEFT method and comes closest to full fine-tuning on Quora and CFIMDB.

Method	Trainable params	Trainable %	Best train loss	Best train ppl	Final token acc
Full fine-tune	125,030,400	100.0000%	1.9456	6.9978	0.5734
LoRA	1,327,104	1.0503%	3.9642	52.6765	0.3374
ReFT	147,456	0.1178%	4.5696	96.5078	0.2905

Table 2: Sonnet generation results. Full fine-tuning is much stronger than LoRA and ReFT on the generative task.

much less and plateaus far below the other two methods. On Quora, the comparison is cleaner: LoRA tracks the full model much more closely than ReFT over all 10 epochs, which matches the strong final performance gap seen in Table 1. Sonnet shows the sharpest separation. Full fine-tuning improves steadily over nearly all 30 epochs, while LoRA rises more slowly and ReFT flattens much earlier.

Figures 4, 5, and 6 tell a similar story from the optimization side. Full fine-tuning reaches the lowest loss on every dataset. On CFIMDB and Quora, the ordering of the curves is stable across training: full fine-tuning is best, LoRA is in the middle, and ReFT remains highest. On SST, however, the full model continues lowering training loss even after dev accuracy has already peaked, which reinforces the overfitting interpretation. The sonnet loss curves are especially informative. Full fine-tuning keeps making steady progress over nearly all 30 epochs, LoRA improves much more slowly, and ReFT drops early and then oscillates around a high-loss plateau. That makes the gap look more like a capacity limitation than a simple optimization issue.

The runtime results in Table 3 add an important practical point. Reducing the number of trainable parameters does not automatically produce a wall-clock speedup. ReFT is faster than full fine-tuning on every task in our runs, sometimes by a large margin. LoRA, however, is not consistently faster and is actually slightly slower than full fine-tuning on SST, CFIMDB, and Quora. This suggests that the full forward pass through the frozen backbone still dominates much of the computation, and that different PEFT implementations carry different overheads.

6 Analysis

Taken together, the results show that PEFT is not a single uniform tradeoff. LoRA is the strongest parameter-efficient method in these experiments because it stays close to full fine-tuning on the classification tasks, especially Quora and CFIMDB. ReFT is more aggressive in reducing trainable parameters and often gives the best runtime, but the quality drop is much larger.

The classification and generation results also differ in a meaningful way. On the label-oriented tasks, relatively small updates appear sufficient to recover much of the benefit of full fine-tuning. This is most obvious on Quora, where LoRA stays within about 1.5 accuracy points of the full model. On sonnet generation, the gap is much larger. That suggests the current PEFT parameter budgets are enough to sharpen a decision boundary but not enough to substantially reshape the model’s generative distribution over many output positions.

The runtime findings are also important. PEFT is often discussed as if fewer trainable parameters should imply faster training, but these runs show that the relationship is more complicated. ReFT

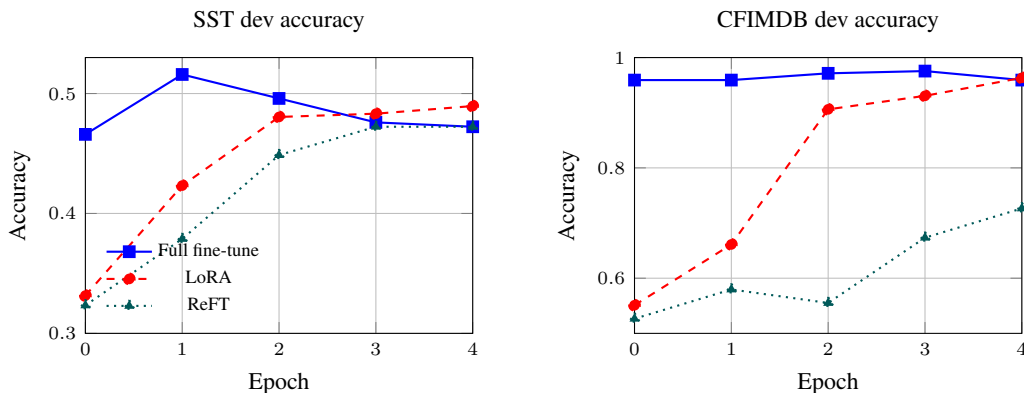


Figure 1: Accuracy curves for the sentiment datasets. The plotted metric is dev accuracy. Full fine-tuning learns fastest, LoRA narrows the gap by the end, and ReFT remains weaker on both datasets.

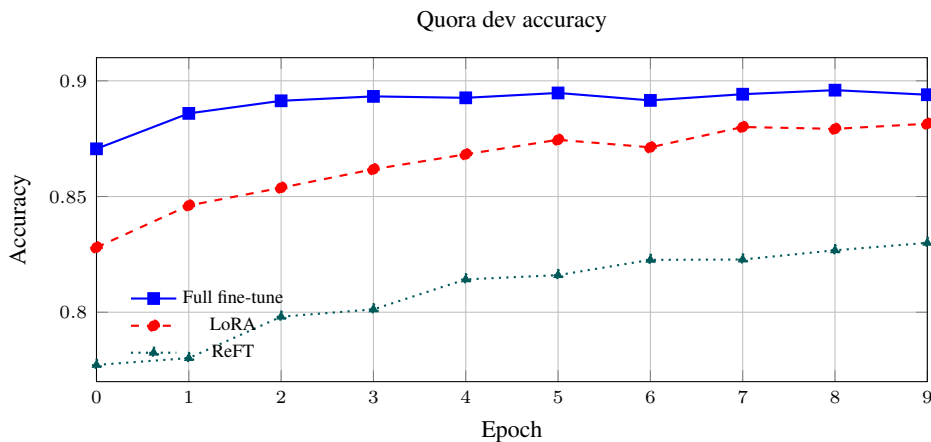


Figure 2: Accuracy curves for Quora paraphrase detection. LoRA tracks full fine-tuning much more closely than ReFT across all 10 epochs.

produces a clear wall-clock improvement in this setup, while LoRA does not. A likely reason is that training cost is still dominated by the frozen GPT-2 forward pass, and adapter-specific implementation overhead matters. This means that PEFT can offer three different tradeoffs at once: parameter count, final quality, and actual training time. The method that is best on one axis need not be best on the others.

There are still several limitations to the current study. Only GPT-2 small is tested, and both PEFT methods use one main rank setting in the final runs. The Quora setup could likely be improved further by scoring only the yes and no tokens rather than effectively competing against the full vocabulary. ReFT may also benefit from more careful intervention placement than the all-layer setting used here, or from higher rank and therefore a higher parameter count specifically for a small model like GPT-2. Finally, a deeper runtime profile that separates forward, backward, and optimizer costs would make the efficiency comparison more precise.

7 Conclusion

This project compared full fine-tuning, LoRA, and ReFT on GPT-2 small across sentiment classification, paraphrase detection, and sonnet generation. LoRA is the strongest PEFT method overall in this setup. It comes close to full fine-tuning on Quora and CFIMDB while training only about 1% of the parameters. ReFT is even more parameter-efficient and often faster, but it gives up substantially more performance. On sonnet generation, both PEFT methods perform much worse than full fine-tuning,

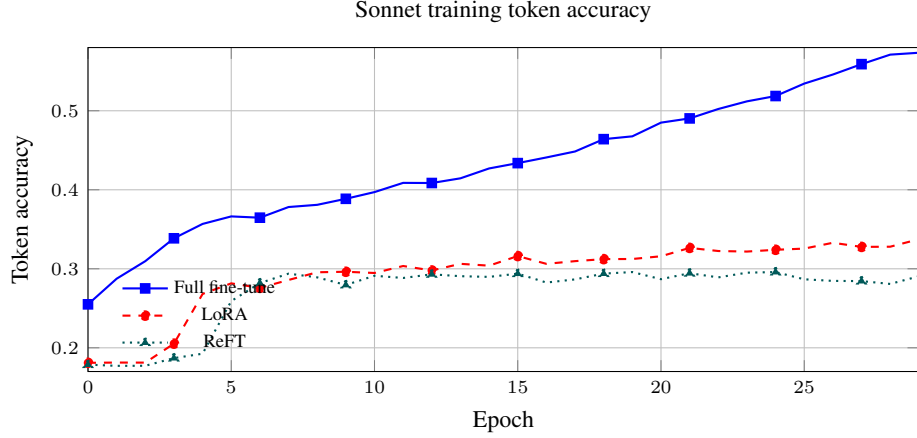


Figure 3: Accuracy curves for sonnet generation. The plotted metric is training token accuracy. Full fine-tuning continues improving across most of training, while LoRA improves more slowly and ReFT plateaus much earlier.

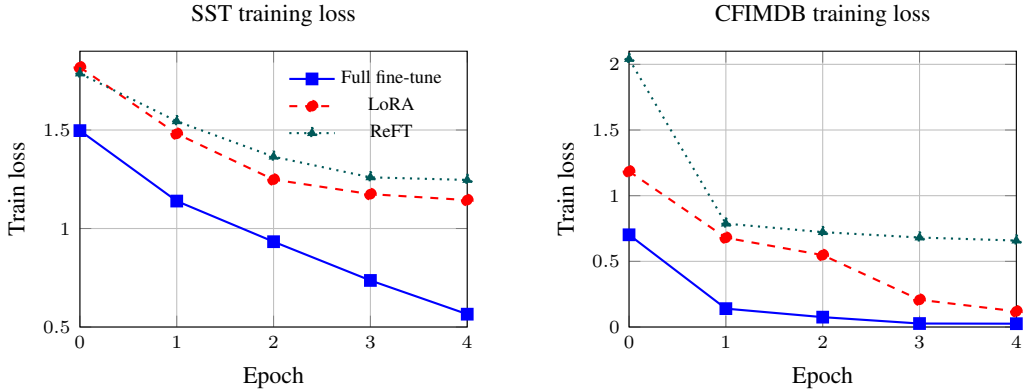


Figure 4: Training loss curves for the sentiment datasets. Full fine-tuning reaches the lowest loss in both cases, while ReFT stalls much earlier at a worse solution.

which suggests that small adaptation modules are more effective for classification than for open-ended generation in this setting. Overall, these results show that PEFT can be a strong alternative to full fine-tuning, but the quality-efficiency tradeoff depends heavily on both the method and the task.

References

- [1] 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. *arXiv preprint arXiv:2106.09685*, 2021.
- [2] Zhengxuan Wu, Aryaman Arora Wang, Zheng Huang, Simon Sun, Wenhao Zhu, Suhas Pai, Robin Jia, Sanjeev Arora, Graham Neubig, Mengzhou Wang, et al. Reft: Representation finetuning for language models. *arXiv preprint arXiv:2404.03592*, 2024.
- [3] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. OpenAI technical report.
- [4] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew 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.

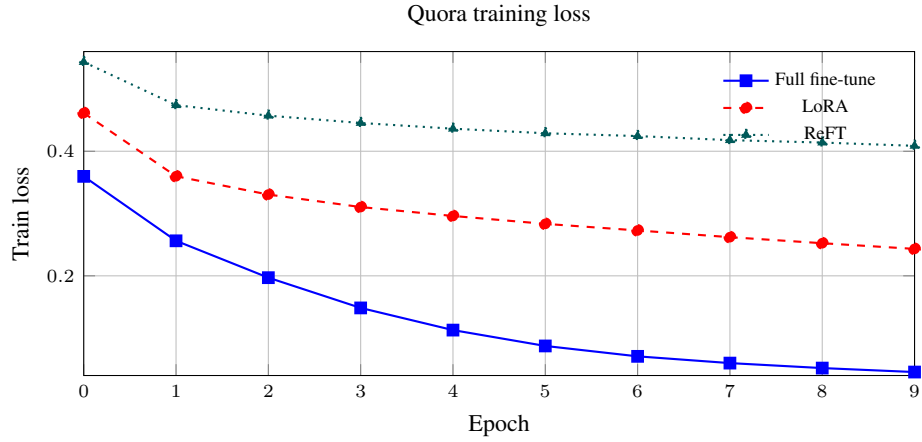


Figure 5: Training loss curves for Quora paraphrase detection. The separation between the three methods is stable throughout training, with LoRA consistently between full fine-tuning and ReFT.

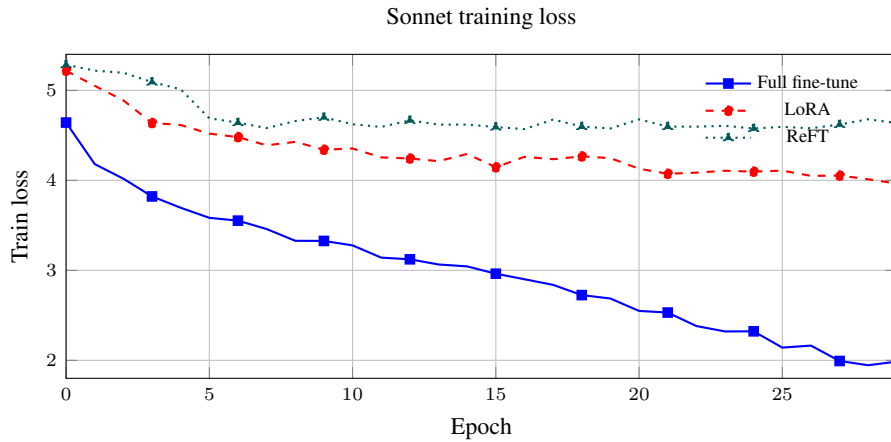


Figure 6: Training loss curves for sonnet generation. Full fine-tuning continues improving across nearly the entire run, while ReFT plateaus early and LoRA improves more gradually.

- [5] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, 2011.
- [6] Stanford NLP. pyreft: Stanford nlp python library for representation finetuning. <https://github.com/stanfordnlp/pyreft>. GitHub repository.
- [7] Hugging Face. Peft: State-of-the-art parameter-efficient fine-tuning. <https://github.com/huggingface/peft>. GitHub repository.

Task	Full (min)	LoRA (min)	ReFT (min)
SST	1.38	1.45	1.00
CFIMDB	2.88	3.07	2.24
Quora	109.65	116.98	75.62
Sonnet	10.83	9.05	1.69

Table 3: Wall-clock training time. ReFT is consistently faster than full fine-tuning in these runs, while LoRA does not provide a reliable speedup.