

Default Final Project: Efficient Adaptation of GPT-2 via LoRA

Stanford CS224N Default Project

Pedro Pires

Department of Computer Science
Stanford University
pfp@stanford.edu

Abstract

This project investigates the implementation and modernization of the GPT-2 (Small) architecture, focusing on parameter-efficient fine-tuning techniques. While large language models exhibit strong zero-shot capabilities, full supervised fine-tuning (SFT) remains computationally prohibitive. We successfully implemented the core GPT-2 architecture and the Adam optimizer from starter code. We then evaluated the model on classification tasks before extending it to Cloze-style paraphrase detection and autoregressive Sonnet generation. To address the computational bottleneck of full fine-tuning, we integrated Low-Rank Adaptation (LoRA) [1]. Our results demonstrate that LoRA achieves competitive performance with our full SFT baselines while reducing peak VRAM usage and trainable parameter counts.

1 Key Information

- Mentor: Ahmed Ahmed
- External Collaborators: None
- Sharing project: No
- Late Days:

Team member	Late days remaining
Pedro Pires	1
Total pooled	1
Team size	1
Late days applied for the report	1

2 Introduction

Generative Pretrained Transformers, such as GPT-2, popularized the "pre-train, then fine-tune" paradigm by demonstrating that large-scale unsupervised pre-training yields strong improvements on downstream NLP tasks. However, updating all parameters of a language model during fine-tuning requires massive computational resources, limiting accessibility for independent researchers.

In this project, we reconstruct the GPT-2 (Small) architecture and investigate parameter-efficient fine-tuning using Low-Rank Adaptation (LoRA). Originally, our scope included alignment via Direct Preference Optimization (DPO) [2] and acceleration via Flash Attention [3]. However, through the course of development, we pivoted to perform an ablation of LoRA's efficiency-performance trade-offs compared to full Supervised Fine-Tuning (SFT). We demonstrate our model's capabilities on two distinct downstream applications: Cloze-style paraphrase detection and constrained Sonnet generation.

3 Related Work

- **GPT-2 and Language Modeling:** GPT-2 is a decoder-only transformer that models text unidirectionally. It showed that scaling autoregressive models improves zero-shot performance and fine-tuning capabilities across diverse tasks.
- **LoRA (Low-Rank Adaptation):** Proposed by Hu et al., LoRA freezes pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture. This significantly reduces the number of trainable parameters for downstream tasks.

4 Approach

We first implemented the core components of the GPT-2 model. This included a masked multi-head self-attention mechanism, which applies an upper-triangular mask to prevent tokens from attending to future positions. We also implemented the complete GPT-2 transformer block, stacking 12 of these layers, and the Adam optimizer featuring decoupled weight decay regularization.

For our downstream tasks:

- **Paraphrase Detection:** We reformulated binary classification into a cloze-style task. The model is prompted to answer whether one sentence is a paraphrase of another by generating the token "yes" or "no".
- **Sonnet Generation:** We utilize autoregressive language modeling. The model is conditioned on the first three lines of a sonnet and must generate the remaining 11 lines, adhering to the strict ABAB CDCD EFEF GG rhyme scheme.
- **LoRA Integration:** Instead of full SFT where all 117M parameters are updated, we freeze the base GPT-2 weights. We inject trainable low-rank matrices A and B into the attention projection layers. The forward pass is modified such that $h = W_0x + \Delta Wx = W_0x + BAx$.

5 Experiments

5.1 Data

We utilize the following datasets:

- **Stanford Sentiment Treebank (SST) & CFIMDB:** Used for initial validation of our architecture. SST contains 11,855 single sentences, and CFIMDB contains 2,434 highly polar reviews.
- **Quora Question Pairs:** For paraphrase detection, we use a subset of the Quora dataset containing 141,506 training examples and 20,215 dev examples.
- **Shakespearean Sonnets:** For generation, we use a dataset of 154 sonnets (143 train, 12 test).

5.2 Evaluation method

For paraphrase detection, we report standard classification **Accuracy**. For sonnet generation, we evaluate the generated text’s closeness to Shakespeare’s original language using the **chrF metric**, a character-level n-gram comparison metric. We also record **Peak VRAM usage** to quantify efficiency gains.

5.3 Experimental details

Our full SFT baseline models were trained using our custom Adam optimizer.

- **Paraphrase Detection (SFT):** Batch size = 8, Learning Rate = 1e-5, Epochs = 10.
- **Sonnet Generation (SFT):** Batch size = 8, Learning Rate = 1e-5, Epochs = 10.
- **LoRA Models:** We set rank $r = 8$ and $\alpha = 32$. The base weights were frozen, LoRA was applied to Q and V attention projections only. We trained with a learning rate of 1e-5.

5.4 Results

Our initial implementation successfully passed baseline requirements, achieving 0.520 Dev accuracy on full-model SST and 0.976 on CFIMDB.

Table 1 details our performance on the core generation tasks comparing Full SFT to LoRA.

Table 1: Performance and Efficiency Comparison: Full SFT vs. LoRA

Task	Method	Metric	Dev Score	Test Score	Peak VRAM (GB)
Paraphrase	Full SFT	Accuracy	0.897	0.831	1.35
Paraphrase	LoRA	Accuracy	0.856	0.814	0.77
Sonnet Gen	Full SFT	chrF	41.834	41.834	1.22
Sonnet Gen	LoRA	chrF	40.490	40.108	0.75

6 Analysis

Our findings indicate that LoRA provides a highly favorable trade-off between task performance and computational cost.

- **Efficiency:** By tracking memory allocation during the backward pass, LoRA reduced our peak VRAM consumption from 1.22 GB to 0.75 GB, allowing for larger batch sizes.
- **Qualitative Sonnet Evaluation:** While both SFT and LoRA learned the basic archaic vocabulary (e.g., "doth", "thou"), but were not able to generate sonnets with the appropriate metric.

7 Conclusion

In this project, we successfully built the GPT-2 architecture and its optimizer from scratch, achieving strong baseline accuracies across sentiment and paraphrase tasks. By integrating LoRA, we demonstrated that modern parameter-efficient techniques can drastically lower the computational barrier for fine-tuning legacy models, with minimal degradation in generative quality or classification accuracy. Future work should revisit integrating Direct Preference Optimization (DPO) to enforce strict rhythmic constraints in sonnet generation through negative sampling, as well as hardware-aware optimizations like Flash Attention.

Team contributions

Pedro Pires completed all aspects of the project, including the implementation of the GPT-2 architecture, Adam optimizer, task baselines, and the LoRA extension.

References

- [1] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR 2022*, April 2022.
- [2] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [3] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359. Curran Associates, Inc., 2022.