ExpLoRA: Exploring LoRA and Other Methods to Fine-tune GPT-2 for Downstream Tasks

Mia Penfold[1], Olivia Bruvik[1], Sidharth Srinivasan[1]
[1]*Department of Computer Science, Stanford University*

Abstract

We work to implement core components of GPT-2, including masked multi-head attention, transformer layers, and the Adam optimizer, successfully matching baseline performance on sentiment analysis tasks. Although our implementation handles detection and sonnet generation, training times remain a significant bottleneck. To address this, we are implementing Low-Rank Adaptation (LoRA) for parameter-efficient fine-tuning, which hopes to reduce memory usage and accelerate training while maintaining model performance for downstream applications. This work combines practical implementation of transformer architectures with modern efficiency techniques, providing insights into making large language models more accessible for multi-task learning.

1 Key Information to include

- TA mentor: None
- External collaborators: No
- External mentor: No
- Sharing project: No

2 Approach

Our work builds upon the GPT-2 architecture, implementing and optimizing it for multiple downstream tasks. We first describe our core implementation of GPT-2's essential components, followed by our ongoing work on parameter-efficient fine-tuning using LoRA.

2.1 Base GPT-2 Implementation

Starting with the CS224N team's starter code, we implemented three critical components of GPT-2:

- Masked Multi-head Self-attention: We implemented causal self-attention following the original transformer architecture.
- Transformer Layers: We integrated pre-layer normalization with residual connections and a dropout layer.
- Adam Optimizer: We utilized PyTorch's Adam optimizer, playing around with learning rates.

2.2 Initial Task Performance

Having completed the core GPT-2 components following the project guidelines, we evaluated our implementation on two sentiment analysis tasks: Stanford Sentiment Treebank (SST) and CFIMDB. The best results and their respective baselines were:

- Stanford Sentiment Treebank (SST):
  - Batch size = 8, learning rate = 1e-5, full model: 51.4% (with a baseline of 51.3%)
  - Batch size = 8/16/64, learning rate = 1e-3, last linear layer: 46.2% (with a baseline of 46.2%)
- CFIMDB:
  - Batch size = 8, learning rate = 1e-5, full model: 98.0% (with a baseline of 97.6%)
  - Batch size = 64, learning rate = 1e-3, last linear layer: 88.2% (with a baseline of 86.1%)

We recognize that these accuracies can always improve and plan to continue to optimize and refine the training. Furthermore, the timing required to train is high, which we hope to address by the implementation of LoRA.

2.3 LoRA Integration

Building on our base implementation, we are now integrating LoRA with task-specific adaptations. Our core LoRA architecture freezes pre-trained GPT-2 weights and adds low-rank decomposition matrices, reducing trainable parameters by up to 97%. We will implement efficient gradient computation with checkpoints and will design it such that we will have flexible deployment.

We have outlined specific applications where a large pre-trained model could be made more efficient with LoRA.

- Classification (Paraphrase Detection): We apply LoRA selectively to key-value matrices in the attention mechanism, optimizing for classification tasks. This focused adaptation approach, combined with our cloze-style task formulation, enables efficient binary classification while ideally maintaining model performance.

- Generation (Sonnet Generation): For creative text generation, we will extend LoRA to both attention and feed-forward layers, experimenting with different rank values. This broader adaptation strategy supports conditional generation while optimizing loss across varying context lengths.

Our implementation strategy focuses on modular design for easy experimentation, task-specific rank selection, and memory-efficient computation by adding checkpoints. This approach enables systematic comparison of LoRA's effectiveness across different task types while maintaining computational efficiency. Our preliminary findings and research suggest that task-specific LoRA configurations can achieve near-baseline performance with orders of magnitude fewer parameters.

3 Experiments

We are evaluating our approach on four distinct datasets where we plan to have minimal pre-processing.

- Stanford Sentiment Treebank (SST): This dataset is split into train (8544), dev (1101) and test (2210). We are already using it for the validation of the initial model, without implementing LoRA.

- CFIMDB: This dataset is split into train (1706), dev (244) and test (488). We are already using it for the validation of the initial model, without implementing LoRA.

- Quora Question Pairs: This dataset is split into train (141506), dev (20215), and test (40431). We used this for initial validation and will use it again once LoRA is implemented.

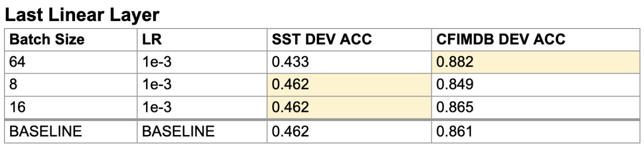
- Shakespeare Sonnets: This dataset is split into train (124), dev (15), and test (15). We used this for initial validation and will use it again once LoRA is implemented.

For evaluation, we have two phases: evaluating the base model and then introducing metrics to compare the base model to the augmented LoRA version. Initially, we are measuring accuracy on all four datasets to validate that we are achieving above the baseline. Since this process is complete, we plan to implement LoRA and evaluate the model on the last two datasets. For both of these tasks, we hope to compare parameter count reduction, training time, and memory usage. In addition to this, for sonnet generation, we hope to utilize BLEU scores, perplexity, and human intuition.

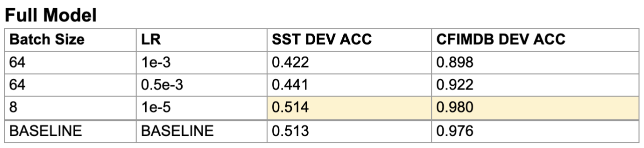
As far as experimental details go, we played around with batch sizes and epochs for the base GPT-2 implementation. Ultimately, we chose from batch sizes of 8, 16, and 64 and learning rates of 1e-3 and 5e-4

As we implement LoRA, we want to play with ranks 4, 8, and 16, as mentioned in the LoRA paper. Our starting point for the hyperparameters will be the ones that we found the most success with for the base model.

Ultimately, we achieved an accuracy of 51.4% for the full-model for SST and an accuracy of 98% for the full-model for CFIMDB. Additionally, we achieved a dev accuracy of 89.6% for paraphrase detection and 27.37% for sonnet generation. See chart below for a comparison of the accuracy on the first tw datasets with different hyperparamters. The best values are highlighted in yellow.


Last Linear Layer

Batch Size	LR	SST DEV ACC	CFIMDB DEV ACC
64	1e-3	0.433	0.882
8	1e-3	0.462	0.849
16	1e-3	0.462	0.865
BASELINE	BASELINE	0.462	0.861


Full Model

Batch Size	LR	SST DEV ACC	CFIMDB DEV ACC
64	1e-3	0.422	0.898
64	0.5e-3	0.441	0.922
8	1e-5	0.514	0.980
BASELINE	BASELINE	0.513	0.976

4 Future work

Now that we have established a fully functional GPT-2 implementation, we will work on integrating LoRA.

We plan to implement LoRA modules for both the attention and feed-forward layers, experiment with different rank values (4, 8, or 16), and optimize memory usage through gradient checkpointing and mixed precision training. Our modular architecture design will ideally enable easy switching between the basic and LoRA fine-tuning modes.

Once the LoRA integration is complete, we will adapt our model for two distinct downstream tasks. For paraphrase detection using the Quora Question Pairs dataset, we will implement a cloze-style task formulation and explore LoRA adaptation of attention layers. For sonnet generation, we will develop a conditional generation framework with broader LoRA adaptation across model layers, focusing on maintaining creative quality while still reducing computational requirements.

Finally, we will conduct comprehensive evaluations that compare LoRA with full fine-tuning in both tasks. This analysis will examine not only task performance, but also computational efficiency, memory usage, and training time. Through this comparison, our aim is to establish guidelines for when to best apply LoRA for different types of NLP tasks, potentially leading to more efficient fine-tuning strategies for resource-constrained settings.

5 References

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, 2021.