

# Fine-tuning GPT-2 with LoRA and DPO for Accurate Classification and Constrained Generation

Stanford CS224N Default Project

**Shaoxiong Zhang**

Department of Computer Science  
Stanford University  
shaoxioz@stanford.edu

## Abstract

Adapting Large Language Models (LLMs) to specialized tasks is frequently hindered by the computational costs of full-parameter fine-tuning and the inability of standard language modeling objectives to enforce strict structural constraints. In this project, we implement a base GPT-2 model from ground up and efficiently adapt it for two distinct downstream tasks: cloze-style paraphrase detection and constrained sonnet generation. To operate over limited GPU resources, we apply Parameter-Efficient Fine-Tuning via Low-Rank Adaptation (LoRA) for the classification task. To enforce the rigid stylistic rules of sonnets without relying on unstable reinforcement learning, we utilize Direct Preference Optimization (DPO). Our experiments demonstrate that these techniques successfully tune the base GPT-2 model toward highly specific objectives, achieving a paraphrase detection accuracy of 0.866 and a sonnet generation chrF score of 42.407.

## 1 Key Information to include

- Mentor: Wei Liu
- External Collaborators (if you have any): No
- Sharing project: No
- Late Days (if you are using any):

| Team member                      | Late days remaining |
|----------------------------------|---------------------|
| <Shaoxiong Zhang>                | <0>                 |
| Total pooled                     | <2>                 |
| Team size                        | <2>                 |
| Late days applied for the report | <2>                 |

## 2 Introduction

The innovative paradigm in modern NLP has shown that pre-trained large-scale models on general-domain data, then adapted via fine-tuning to specific downstream tasks, can often achieve state-of-the-art performance. However, there are challenges as the model scale grows with the scaling law. Full-parameter fine-tuning becomes resource-intensive, requiring tremendous GPU compute and memory. Also, the standard cross-entropy loss objective frequently fails to enforce strict stylistic and structural constraints.

In this project, we implement a vanilla GPT-2 model and extend it to tackle two unique downstream tasks. First task is paraphrase detection to identify whether two sentences are semantic equivalent, we reformulate it into a cloze-style task to prompt the model generate an answer of yes or no. Due to the scale of training data and the limited GPU resources we have, full-parameter fine-tuning is less

preferred. We address this issue using Parameter-Efficient Fine-Tuning via Low-Rank Adaptation (LoRA) [1]. By freezing most of the pre-trained weights and injecting parallel trainable rank decomposition matrices to reduce the resource footprint and training time while keeping the model representation capability.

Second task is sonnet generation, which requires the outputs not just to make semantic sense but also to apply to Shakespeare’s style, which has a strict 14-line structure and an ABAB CDCD EFEF GG rhyme scheme. We use Direct Preference Optimization (DPO) [2] with synthetic winning losing pairwise data to align the model generation with structural preferences. It acts as an implicit reward model to penalize structural failures and bypasses the instability of complex reinforcement learning pipelines.

Finally, our experiments demonstrate that these two extensions on the base model are effective to the downstream tasks. We achieved a paraphrase detection accuracy of 0.866 and a sonnet generation chrF [3] score of 42.407.

### 3 Related Work

In this section, we discuss and compare different approaches to solve the cloze-style paraphrase detection and constrained sonnet generation.

Paraphrase detection has traditionally relied on lexical semantic similarity to determine if two phrases convey the same semantic meaning, as demonstrated in [4]. However, we pivot by reformulating the classification task as cloze-style generation to leverage the contextual understanding of pre-trained large language models. To fine-tune the base model under limited GPU resources, we leverage LoRA, which freezes the pre-trained weights and injects trainable rank decomposition matrices that significantly reduce the total trainable parameters. We’ve compared several alternative methods and outlined the reasons for not selecting these options. Representation Finetuning (ReFT) [5] operates on hidden representations rather than model weights; it’s novel, but LoRA is more established and easier to implement than deploy. Quantization (GPTQ / QuAdapter) reduces memory by representing weights in lower precision, like int8. We intentionally avoid this option to prevent potential degradation in semantic interpretation and instead using LoRA’s high-fidelity low-rank efficient updates. SMART [6] Regularization uses smoothness-inducing adversarial regularization to prevent overfitting during fine-tuning. We didn’t take this complex optimization because LoRA’s low-rank bottleneck inherently already acted as a sufficient structural regularizer. Second-Order Optimizers (Shampoo, SOAP) precondition the gradient with higher-order terms to speed up convergence, but computing these preconditioners adds significant computational overhead, and our bottleneck is not about training convergence. Similar to FlashAttention [7] utilizes CUDA optimizations to compute exact attention faster to speed up training, we didn’t use that because our primary bottleneck is parameter count and memory rather than long-sequence computation.

Sonnet generation, enforcing strict rhyme and stylistic constraints, reveals the limitations of standard cross-entropy supervised fine-tuning. We leverage DPO to align large language models with human preferences using pairwise comparison data. By constructing winning and losing data, contrasting correctly formatted sonnets against corrupted ones, we treat structural rules and stylistic outputs as generation preferences. Reinforcement Learning from Human Feedback (RLHF) relies on separate reward models and is usually unstable during training compared to DPO’s supervised approach. Kronecker-Factored Approximate Curvature (K-FAC) [8] is an efficient method for approximating natural gradient descent in neural networks. It relies on efficient invertible approximation of the network’s Fisher information matrix, where large blocks corresponding to layers are approximated as the Kronecker product of two smaller matrices. We didn’t take this option, although K-FAC can optimize the objective much faster than standard stochastic gradient descent, because our main focus is on preference learning. Regarding to the evaluation metrics, traditional word-level evaluation metrics fall short in capturing the granular structural and stylistic rules. We use the chrF metric to calculate character-level n-gram F-score to accurately measure the model output’s adherence to the strict sonnet generation constraints.

## 4 Approach

In this section, we detail our approach to solve two distinct downstream tasks cloze-style paraphrase detection and constrained sonnet generation by extending the base GPT-2 model architecture. We begin with self-implemented GPT-2 baseline; then illustrate how we overcome the GPU resource constraints and long training time by replacing full-parameter fine-tuning with LoRA during paraphrase detection; finally discuss the staged DPO pipeline to enforce strict structural and stylistic constraints for sonnet generation.

### 4.1 Baseline Model

Our baseline is a decoder-only Transformer based on GPT-2 architecture. Self-implemented from ground up as part of the CS224N default project part 1, the architecture consists of 12 transformer layers, each utilizing masked multi-head self-attention and a hidden dimension of 768. GPT-2 base model is pre-trained via standard auto-regressive language modeling, with the optimization objective maximizing the log-likelihood to predict the next token  $x_i$  over a sequence of length  $n$ :

$$\log P(x_1, x_2, \dots, x_{i-1}) = \log \prod_{i=1}^n P(x_i | x_1, x_2, \dots, x_{i-1}) = \sum_{i=1}^n \log P(x_i | x_1, x_2, \dots, x_{i-1}) \quad (1)$$

The goal is to teach the model general language understanding and give it the ability to generate text with semantic meaning. The model can then be fine-tuned later to perform downstream tasks like paraphrase detection and sonnet generation.

### 4.2 Cloze-Style Paraphrase Detection with Low-Rank Adaption

Rather than modifying the baseline architecture to append a separate sequence classification head, we reformulate the binary paraphrase detection task into a generative cloze-style objective. The model receives a prompt formatted as: "Is 'Sentence 1' a paraphrase of 'Sentence 2'? Answer 'yes' or 'no': " and is restricted to generating a single-token output.

To optimize the cloze-style generative classification under limited GPU resources, we implement LoRA. We freeze the weights of GPT-2 pre-trained layers and inject trainable low-rank decomposition matrices in parallel to the existing linear layers. Specifically, we only attach LoRA modules to the attention layer query ( $W_q$ ) and value ( $W_v$ ) projection matrices within each transformer block, as this setting results in an optimal performance-to-parameter ratio.

For a pre-trained weight matrix  $W_0 \in \mathbb{R}^{d \times k}$ , we represent its update  $\Delta W$  with a low-rank decomposition:

$$h = W_0 x + \Delta W x = W_0 x + B A x \quad (2)$$

where  $B \in \mathbb{R}^{d \times r}$ ,  $A \in \mathbb{R}^{r \times k}$ , and the rank  $r \ll \min(d, k)$ .

The whole LoRA injection and forward function is original work and implemented by me.

### 4.3 Constrained Sonnet Generation with Direct Preference Optimization

For sonnet generation, the model will be conditioned on the first 3 lines of the sonnets and generate the remaining 11 lines. We optimize this generation in three phases:

- **Phase 1, Supervised Fine-Tuning (SFT).** We first apply a full-parameter fine-tuning on the pre-trained GPT-2 model on the positive sonnet samples using the standard cross-entropy loss ( $\mathcal{L}_{LM}$ ). The reason why we need SFT is to anchor the model into a stable state to minimize the KL-divergence gap during DPO training. The SFT model will be frozen during DPO.
- **Phase 2, Pairwise Data Construction.** The default sonnet dataset implementation only contains positive samples. We will construct the synthetic pairwise data with winning and

losing samples  $(x, y_w, y_l)$ , where the winning sample ( $y_w$ ) is the true sonnet and the losing sample is ( $y_l$ ) that was generated from heuristic modifications like:

- Breaking the rhymes like randomly replacing 1 word with a pre-defined irrelevant word.
  - Repeat lines, reverse lines to break the line number restriction.
  - Randomly shuffle the lines.
  - Randomly truncate characters.
- **Phase 3, Direct Preference Optimization (DPO).** As we discussed in the related work section, we chose DPO to enforce output structural preference without unstable reinforcement learning. The model architecture will be updated for DPO. Two identical copies of the SFT model will be loaded into memory: a frozen reference model ( $\pi_{ref}$ ) and a policy model ( $\pi_\theta$ ) that updates weights with the AdamW optimizer. The crucial part is computing the sequence log-probabilities for the DPO loss, where  $\sigma$  is the sigmoid function and  $\beta$  is the temperature parameter controlling the preference signal strength:

$$\mathcal{L}_{DPO} = -\mathbb{E}_{(x, y_w, y_l) \sim D} \left[ \log \sigma \left( \beta \log \frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)} \right) \right] \quad (3)$$

We apply an attention mask to isolate the completion tokens strictly ( $t \geq \text{prompt\_len}$ ). This ensures that the model loss is rewarded or penalized for its generated continuation that skips the first 3-line conditioning context. We then update the policy model  $\pi_\theta$  to increase the likelihood of the correctly structured sonnet while penalizing the corrupted generation. We also apply gradient clipping during the backward pass to stabilize the dual-model alignment update process.

The whole three-phase DPO pipeline is original and implemented by me.

## 5 Experiments

To assess our LoRA and DPO fine-tuned GPT-2 models, this section outlines the data utilized for each task, the evaluation method to measure the performance, the experimental details during training and eval, and the final quantitative results.

### 5.1 Data

We utilize the same datasets provided in the default project handout and the code base:

- **Cloze-Style Paraphrase Detection.** A subset of the Quora dataset contains question pairs. The dataset processor will reformulate this into a generative task with a cloze-style input prompt: "Is 'Sentence 1' a paraphrase of 'Sentence 2'? Answer 'yes' or 'no': ". The model is restricted to generating a single-token "yes" or "no" target output.
- **Constrained Sonnet Generation.** A collection of 154 Shakespeare’s style sonnets. In DPO, a corrupted dataset will be generated with heuristic rules from the positive samples. During testing, the model will be conditioned on the first 3 lines of each sonnet and generate the remaining 11 lines.

### 5.2 Evaluation method

My evaluation method is the same as the default project instructions:

- **Cloze-Style Paraphrase Detection.** The classification accuracy on the test set predictions will be used to evaluate the model’s ability to identify the semantic equivalence.
- **Constrained Sonnet Generation.** The chrF (character n-gram F-score) metrics are used to quantitatively measure the stylistic fidelity and evaluate how closely the generated text matches the character-level language distribution of original positive sonnets.

### 5.3 Experimental details

In this section, we discuss the experimental details the two tasks separately. For paraphrase detection, we focus on how LoRA improve efficiency on fine-tuning in terms of trainable parameters, training time, and GPU memory footprint with comparable performance. For sonnet generation, we focus on how DPO help align the model outputs with preferred style to improve the evaluation metric scores. Both tasks are trained on GCP with a G4 with Nvidia RTX PRO 6000 spot instance with 96GB memory and 120 FP32 TFLOPS, the conda environment has to be updated to python 3.10 to get the GPU driver working.

#### 5.3.1 Paraphrase Detection

- **Model Configuration.** The base implementation is pre-trained GPT-2 architecture, we first run the baseline with model size *gpt2* without LoRA. Then adding LoRA layers for query/value matrices and froze the base SFT model parameters. The task is formulated as sequence classification by extracting the hidden state of the final token and passing it through a binary linear classification head.
- **Training Config.** In baseline implementation, we fully fine-tuned the standard GPT-2 model using AdamW with standard cross-entropy loss. To improve training efficiency, we introduced LoRA. We swept through different learning rates (1e-5 to 1e-3), explored various LoRA ranks ( $r = 1, 2, 4$ ), and compared base model sizes (GPT-2 vs. GPT-2 Medium) to achieve the best performance for paraphrase detection.

#### 5.3.2 Sonnet Generation

- **Model Configuration.** Different with the paraphrase detection, the dataset is much smaller and we focus on maximize the chrF evaluation metrics in this experiment. The base implementation is the standard full-parameter SFT over pre-trained GPT-2 architecture, then the whole sequence hidden states will be used to decode using top-p (nucleus) generation algorithm. Then we improve the base implementation with DPO, where the policy model is trained with DPO loss to assign higher probabilities to Shakespeare’s style and lower probabilities for corrupted losing samples.
- **Training Config.** In base implementation the model is trained using AdamW with standard cross-entropy loss, lr 3e-5 and weight decay 0.01 with batch size of 8; for generation algorithm hyper parameters, *top\_p* = 0.9 and *temperature* = 1. In DPO training, the batch size is set to half of the base SFT as we will have both winning and losing samples in a single batch item. We first tried different model size on the base SFT to make sure the model is anchored to the good state before DPO. Then we tried different learning rate, and different beta to get the best performance. Then we remove certain corrupted strategies in DPO dataset and then see if the chrF score is decreased.

### 5.4 Results

For paraphrase detection, we also provide a diagram compare the resource utilization besides the training results table.

#### 5.4.1 Paraphrase Detection

Please see Table 1 for training results for cloze-style paraphrase detection, and Figure 1 for efficiency improvements of LoRA fine-tuning over full-parameter fine-tuning in terms of training time and GPU memory utilization.

#### 5.4.2 Sonnet Generation

Please see Table 2 for training results for constrained sonnet generation.

| Setup           | Model Size  | LR   | Dev Acc | Trainable Params | Training Time (mins) |
|-----------------|-------------|------|---------|------------------|----------------------|
| Baseline        | N/A         | N/A  | 0.882   | N/A              | N/A                  |
| Full-params SFT | gpt2        | 1e-5 | 0.892   | 125M             | 62                   |
| LoRA-lr         | gpt2        | 1e-5 | 0.823   | 296K             | 46                   |
|                 | gpt2        | 1e-4 | 0.865   | 296K             | 46                   |
|                 | gpt2        | 1e-3 | 0.87    | 296K             | 46                   |
|                 | gpt2        | 5e-4 | 0.877   | 296K             | 46                   |
| LoRA-model-size | gpt2-medium | 1e-4 | 0.889   | 788K             | 128                  |
| LoRA-rank-4     | gpt2        | 5e-4 | 0.869   | 296K             | 46                   |
| LoRA-rank-2     | gpt2        | 5e-4 | 0.845   | 296K             | 46                   |
| LoRA-rank-1     | gpt2        | 5e-4 | 0.650   | 296K             | 46                   |

Table 1: Cloze-style paraphrase detection experiment results

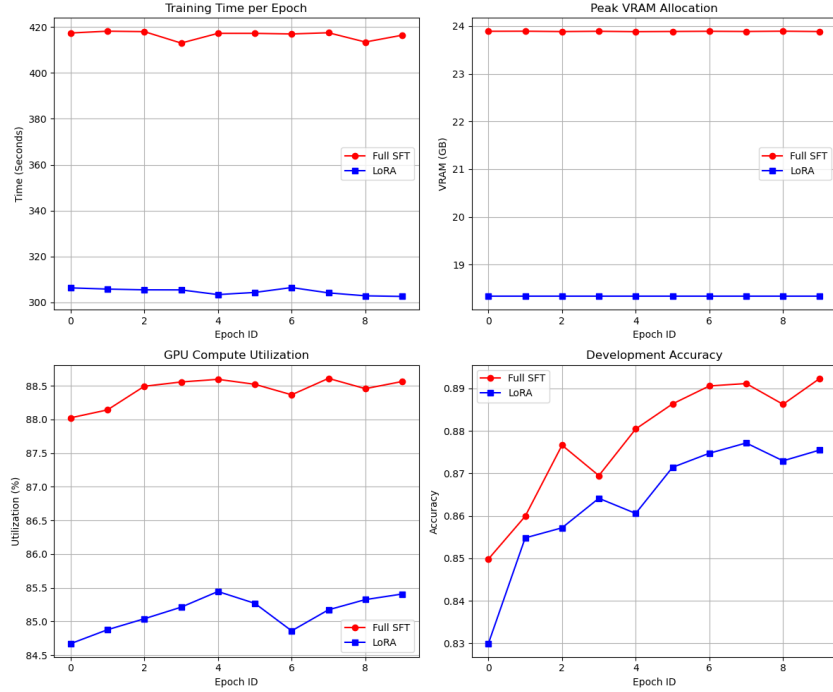


Figure 1: LoRA V.S. SFT on GPT-2 fine-tuning

## 6 Analysis

### 6.1 Paraphrase Detection: Efficiency and Quality Trade-offs

Our qualitative and quantitative evaluations of the paraphrase detection task reveal a distinct trade-off between computational efficiency and model capacity. As illustrated in Figure 1, LoRA significantly reduces the hardware burden compared to full-parameter SFT. Peak VRAM allocation drops from roughly 23.9 GB to 18.3 GB, and training time per epoch is reduced by approximately 25%.

Despite freezing the vast majority of the network, LoRA with  $r \geq 4$  recovers nearly 98% of the baseline SFT accuracy (0.877 vs. 0.892). However, our ablation study on the LoRA rank parameter (Table 1) exposes a strict capacity bottleneck. While reducing the rank from 4 to 2 causes a moderate accuracy drop (to 0.845), further compressing the rank to 1 results in a severe drop to 0.650. This indicates that identifying semantic equivalence requires a minimum degree of representational capability in the attention query and value projections; a rank of 1 simply lacks the bandwidth to map complex sentence embeddings to a reliable binary classification.

| Setup        | Model Size  | DPO LR | DPO Epochs | DPO Beta | Dev chrF |
|--------------|-------------|--------|------------|----------|----------|
| Baseline     | N/A         | N/A    | N/A        | N/A      | 41.41    |
| SFT          | gpt2        | N/A    | N/A        | N/A      | 42.39    |
|              | gpt2-medium | N/A    | N/A        | N/A      | 41.04    |
|              | gpt2-large  | N/A    | N/A        | N/A      | 41.86    |
| SFT+DPO      | gpt2        | 5e-6   | 5          | 0.1      | 40.50    |
|              | gpt2        | 5e-6   | 2          | 0.1      | 43.11    |
|              | gpt2        | 1e-5   | 2          | 0.5      | 43.46    |
| SFT+DPO-lite | gpt2        | 1e-5   | 2          | 0.5      | 40.90    |

Table 2: Constrained sonnet generation experiment results

## 6.2 Sonnet Generation: Model Scale and Preference Alignment

Our analysis of the constrained sonnet generation task yields counterintuitive results regarding model scale. Surprisingly, the standard *gpt2* architecture outperformed both *gpt2-medium* and *gpt2-large* during the initial SFT phase. This is likely due to the extremely small size of the training dataset, which comprises fewer than 200 sonnets. The smaller model’s reduced capacity inherently acts as a regularizer, making it much less prone to overfitting on this small dataset than the larger variants.

Furthermore, the experiments highlight that establishing a strong SFT baseline is a critical step for preference learning. SFT anchors the model into a good state. Once anchored, DPO can successfully enforce rigid structural rules to maximize the chrF score. However, DPO is highly sensitive to hyperparameter choices on such small datasets. We observed that applying a higher beta penalty (0.5 instead of 0.1) improved the chrF score significantly by keeping the model’s policy closer to the reliable SFT reference. Finally, limiting DPO training to just 2 epochs yielded far better results than 5 epochs (43.46 vs. 40.50), confirming that extended preference optimization on a dataset of this size rapidly leads to overfitting and degrades generation quality.

## 7 Conclusion

In this project, we successfully extended a baseline GPT-2 model using parameter-efficient fine-tuning and preference learning to tackle two distinct downstream tasks. By implementing LoRA, we adapted the model for cloze-style paraphrase detection with a significantly reduced memory footprint, demonstrating that low-rank updates are highly effective for accurate classification. Furthermore, we utilized DPO for structurally constrained sonnet generation. By framing structural and stylistic rules as a preference optimization problem, we guided the model to adhere to Shakespearean constraints without the need for a complex and unstable RLHF pipeline.

A primary limitation of this work stems from the computational cost and extended training times associated with the large-scale paraphrase detection dataset. Consequently, I was unable to perform an exhaustive hyperparameter sweep to figure out the most optimal configuration for learning rate, model size, and weight decay. Additionally, crafting highly effective synthetic negative samples for DPO proved challenging, as manual heuristic corruptions may not perfectly capture the nuanced ways a model generates outputs. For future work, we could explore dynamic rank allocation for LoRA to optimize the capacity-efficiency trade-off more granularly.

## 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. *International Conference on Learning Representations (ICLR)*, 2022.
- [2] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Neural Information Processing Systems (NeurIPS)*, 2023.
- [3] Maja Popović. chrF: character n-gram Fscore for automatic MT evaluation. In *Proceedings of the 10th Workshop on Statistical Machine Translation*, 2015.

- [4] S. Fernando and M. Stevenson. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*, 2008.
- [5] Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. Reft: Representation finetuning for language models. In *Advances in Neural Information Processing Systems*, 2024.
- [6] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.
- [7] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 2022.
- [8] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, 2015.