

# Parameter-efficient Finetuning and Preference Optimization of GPT-2 for Downstream Tasks

Stanford CS224N {Default} Project

**Pankaj Rajak**

Department of Computer Science  
Stanford University  
prajak7@stanford.edu

## Abstract

Pre-trained large language models (LLMs) exhibit strong general language understanding, yet task-specific fine-tuning remains computationally prohibitive at scale, with traditional approaches requiring updates to all model parameters and imposing extreme memory and storage demands. Parameter-efficient fine-tuning (PEFT) methods have emerged to address this bottleneck, though approaches like adapter layers and prefix-tuning carry notable drawbacks including inference latency and optimization instability. We investigate a two-stage fine-tuning approach combining Low-Rank Adaptation (LoRA) with Direct Preference Optimization (DPO) for aligning GPT-2 across three diverse tasks: sentiment analysis, paraphrase detection, and sonnet generation. In the first stage, LoRA-based supervised fine-tuning efficiently adapts the pretrained model, and in the second stage, DPO-based preference alignment refines the model using the SFT checkpoint as a reference. Our experiments show that LoRA consistently matches or surpasses full fine-tuning while providing over 2x training speedup, with optimal configurations being task-dependent — discriminative tasks benefit from higher ranks for greater adaptation capacity, while generative tasks favor minimal ranks to preserve base model fluency. Learning rate selection proves critical, with LoRA requiring roughly 10x higher rates than full fine-tuning, while batch size shows minimal impact. For the DPO stage, moderate regularization is essential to prevent policy drift, with both preference strength and regularization coefficient requiring task-specific tuning. The best LoRA configuration surpasses full fine-tuning on test accuracy for paraphrase detection, and DPO with appropriate regularization yields meaningful improvements over supervised fine-tuning alone on both tasks.

## 1 Key Information to include

- Mentor: Chenglei Si
- External Collaborators (if you have any): No
- Sharing project: No
- Late Days (if you are using any): 2 day

## 2 Introduction

The rapid scaling of large language models (LLMs) has delivered remarkable capabilities across natural language understanding, generation, and reasoning tasks. Modern LLMs range from 1 billion parameters in optimized edge models like Llama 3.2 to 405 billion in flagship systems like Llama 3.1, with some open-source families spanning from 1 billion to 2 trillion parameters. While this

growth in scale drives substantial performance gains, it creates a critical bottleneck: the prohibitive cost of adapting these models to specific downstream tasks through conventional full fine-tuning. Full fine-tuning updates every parameter in the base model and demands massive GPU memory. For example, a 7 billion-parameter model requires approximately 60 GB of VRAM, while a 13 billion-parameter model needs 120 GB or more. These requirements place full fine-tuning beyond the reach of most research labs, small enterprises, and individual practitioners, concentrating LLM adaptation capabilities among a handful of well-resourced organizations.

Parameter-Efficient Fine-Tuning (PEFT) addresses these challenges by enabling effective task adaptation using a minimal set of tunable parameters—often less than 1.0% of the original model size—without significant loss in accuracy or robustness relative to full-model tuning [1]. PEFT approaches freeze most pretrained parameters while fine-tuning only a small number of additional or selected model parameters, thereby greatly decreasing computational and storage costs. This selective updating overcomes catastrophic forgetting, a well-documented behavior observed during full fine-tuning of LLMs where the model loses previously learned knowledge [2]. The democratization potential of PEFT is substantial. Hugging Face demonstrated that fine-tuning 20 billion-parameter LLMs with reinforcement learning from human feedback (RLHF) on a single 24 GB consumer GPU is possible using PEFT techniques such as LoRA [3]. Methods like QLoRA further compress memory requirements in comparison of full fine-tuning [4]. This dramatic reduction in resource requirements opens the door for a far broader community of researchers and developers to customize state-of-the-art models for their specific needs.

Beyond computational efficiency, aligning LLMs with human preferences has become an equally critical research direction. Preference optimization methods such as Direct Preference Optimization (DPO) offer streamlined alternatives to the complex RLHF pipeline [5]. The combination of PEFT techniques with preference optimization creates a powerful and accessible framework for building safe, helpful, and aligned language models.

### 3 Related Work

The PEFT literature encompasses a diverse family of methods, each offering distinct trade-offs between parameter efficiency, performance, and implementation complexity. Understanding this landscape is essential for selecting the appropriate technique for a given application. Adapter-based methods represent one of the earliest PEFT paradigms [6]. Adapters insert small feed-forward modules after attention or feed-forward network layers within each transformer block, while the base model remains entirely frozen. Typically requiring only a few million trainable parameters (roughly 1 – 3% of the total), adapters achieve near full fine-tuning performance but introduce slight inference latency due to the added computational modules. Prefix tuning takes a different approach by prepending trainable prefix vectors to the keys and values at each transformer layer [7]. The base model stays frozen while only the prefixes are optimized, training roughly 0.1% of the parameters. This method offers good performance with very few parameters, though it can be limited in expressiveness for highly complex tasks. A related but simpler technique, prompt tuning, prepends soft learnable tokens to the input embeddings only at the first layer rather than at every layer [8]. Training fewer than 0.1% of parameters, prompt tuning is the most cost-effective approach, particularly for large models, though it generally has a lower performance ceiling. BitFit takes a minimalist approach by tuning only the bias terms of the pre-trained model, affecting approximately 0.1% of parameters [9]. It offers an attractive balance between performance and parameter efficiency and has been highlighted as a promising alternative for resource-constrained or streaming task settings. Among these methods, Low-Rank Adaptation (LoRA) has emerged as a particularly effective approach [3]. A 2025 comparative study on text classification found that LoRA consistently achieved the highest F1 scores (0.909 on Amazon Reviews), while newer methods like ReFT [10] delivered approximately 98% of LoRA’s performance while training only about 3% of the parameters [11]. LoRA occupies a favorable middle ground, training approximately 0.5 – 5% of parameters while consistently delivering top-tier performance.

Aligning LLMs with human preferences is a critical step in producing safe and useful models. The dominant paradigm, Reinforcement Learning from Human Feedback (RLHF) [12], involves training a separate reward model on human preference data and then optimizing the language model policy using proximal policy optimization (PPO). While effective, RLHF requires meticulous hyperparameter tuning and extensive computational resources to maintain RL training stability [13]. Research has

also identified challenges such as reward hacking, reward misspecification, and out-of-distribution generalization issues with explicit reward models [13]. Direct Preference Optimization (DPO) emerged as a landmark alternative, eliminating the need for a separate reward model by deriving a closed-form solution directly from the RLHF objective [5]. DPO requires paired preference data in the form of chosen and rejected responses for each prompt. Since its introduction, several variants have been proposed to address specific limitations or extend its applicability. Identity Preference Optimization (IPO), introduced by Google DeepMind, adds a regularization term to the DPO loss that prevents overfitting and enables training to convergence without requiring tricks like early stopping [14]. In paired preference settings, IPO performs on par with DPO. Kahneman-Tversky Optimization (KTO) departs from the paired preference requirement entirely, defining its loss function in terms of individual examples labeled as good or bad [15]. This makes KTO particularly suitable for production environments where only thumbs-up/thumbs-down signals are available, and it maintains superior performance in scenarios where DPO cannot be applied due to the absence of paired data.

## 4 Approach

### 4.1 Low-Rank Adaptation (LoRA) for Parameter-Efficient Fine-Tuning

We use Low-Rank Adaptation (LoRA) as our primary PEFT method due to its favorable balance of parameter efficiency, performance, practical simplicity and benefits on low-data regime during fine-tuning. The core insight behind LoRA is that the weight updates during fine-tuning occupy a low-rank subspace, meaning that the adaptation can be effectively captured by low-rank matrix decompositions rather than full-rank updates. LoRA introduces two trainable matrices  $A \in \mathbb{R}^{d \times r}$  and  $B \in \mathbb{R}^{r \times d}$  alongside a frozen pretrained weight matrix  $W_0 \in \mathbb{R}^{d \times d}$ , where the rank  $r$  is typically small (2–64) and much smaller than the model dimension  $d$ . The modified forward pass becomes  $h = W_0x + \frac{r}{\alpha}BAx$ . Here,  $\alpha$  is a hyper-parameter and together  $\frac{r}{\alpha}$  defines a scaling factor. This formulation drastically reduces the number of trainable parameters from  $d^2$  to  $2dr$ , which for typical configurations represents a reduction of several orders of magnitude. Here, matrix  $A$  is initialized with random Gaussian values, while matrix  $B$  is initialized as a matrix of complete zeros. This ensures that the product  $\Delta W = BA = 0$  at the start of training, so the model begins from the exact pretrained state, and the LoRA matrices do not perturb the output in a random fashion from the very beginning.

### 4.2 Direct Preference Optimization

Direct Preference Optimization (DPO) represents the next stage after LoRA-based parameter-efficient fine-tuning, offering a mathematically elegant approach to aligning models with human preferences that matches traditional RLHF in quality while dramatically simplifying the process. DPO’s key innovation is eliminating three computationally expensive RLHF components: the separate reward model training phase, the language model sampling during fine-tuning, and the complex PPO algorithm with its actor-critic architecture. This streamlined approach delivers stable, high-performance alignment with minimal computational overhead and significantly reduces the hyperparameter tuning burden that typically accompanies RLHF implementations. The objective function for DPO is given below:

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

Here, DPO objective is a simple binary cross-entropy loss over preference pairs  $(y_w, y_l)$  that optimizes the log-ratio difference, where the implicit reward is parameterized by the policy model  $\pi_\theta$  itself.  $\pi_{\text{ref}}$  is the reference policy, normally it is pre-trained model or supervised fine tuned model. Here,  $\beta$  is the regularization term and a hyperparameter, controlling how much we want  $\pi_\theta$  to deviate with respect to  $\pi_{\text{ref}}$ .  $y_w$  and  $y_l$  is winning and losing response for input  $x$ .

### 4.3 Two-Stage Fine-tuning: LORA based SFT + DPO alignment

In this work, we use the pre-trained GPT-2 model as our foundation. The fine-tuning workflow follows a two-stage process: first, supervised fine-tuning (SFT) with LoRA is applied to the pre-trained model; second, DPO alignment with LoRA is performed on a preference dataset, using the SFT-trained model as the reference model. Throughout both the SFT and DPO stages, LoRA is

applied to all key, value, and query matrices in every attention layer. During the DPO stage, an additional regularization term is added, which is task specific loss function same as what is used during SFT. Thus, the complete loss function during DPO stage is as given below, where  $\lambda$  is the hyperparameter and controls the strength of SFT loss.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{DPO}} + \lambda \mathcal{L}_{\text{Task-specific-sft-Loss}}$$

As a baseline, we perform full fine-tuning of all GPT-2 layers.

## 5 Experiments

We experimented with three different tasks which are sentiment analysis, paraphrase detection and sonnet generation.

### 5.1 Data

For sentiment analysis, we use two movie review datasets: the Stanford Sentiment Treebank (SST) dataset and the CFIMDB dataset. The SST dataset consists of single sentences from movie reviews with five output labels: negative, somewhat negative, neutral, somewhat positive, or positive. The CFIMDB dataset consists of binary labels (positive or negative) for each movie review. For paraphrase detection, we use the Quora dataset consisting of 400,000 question pairs with labels indicating whether instances are paraphrases of one another. Finally, for sonnet generation, we use the Sonnet Dataset consisting of 154 sonnets written by Shakespeare. Each sonnet consists of 14 lines of text following the rhyme scheme ABAB CDCD EFEF GG, where lines assigned the same letter rhyme with each other. For sonnet generation, we also constructed an instruction fine-tuning dataset using data augmentation techniques. Here, an input prompt explains the rhyme scheme and the task the model needs to perform: either generate a shakespeare sonnet or complete a sonnet given the first three lines. See the appendix A for details.

To generate the preference dataset, we used the following approach. For paraphrase detection, we simply flipped the label to create the losing sample for each training example. For sonnet generation, we use two approaches to generate losing samples: (a) a heuristic-based approach where we randomly shuffled lines in the input sonnet, and (b) taking the first three lines from the input sample and using the reference SFT-trained model to complete the remaining 11 lines.

### 5.2 Evaluation method

For both sentiment analysis and paraphrase detection, we used accuracy of the models on held out test dataset as evaluation criteria as they are binary or multi-class classification tasks. In case of Sonnet generation, we used chrF score on held out test dataset as evaluation criteria. chrF score character-level n-gram comparison metric similar to BLEU and is a good measure for language generation tasks.

### 5.3 Experimental details

**Model Architecture:** We use the pre-trained GPT-2 model in our experiments. The GPT-2 model has 12 transformer layers with an embedding dimension of 768 and 12 attention heads per layer. This configuration, known as GPT-2 Small, contains approximately 125 million parameters and serves as the foundation for our parameter-efficient fine-tuning experiments with LoRA. During the LoRA fine-tuning stage, we tune only the key, query, and value projection matrices across all 12 transformer layers, with the LoRA rank varied between 2 and 32. The LoRA and DPO based fine-tuning is only performed for the paraphrase detection and sonnet generation. Here, all models are trained using Adam Optimizers.

For sentiment analysis, we added a fully connected linear layer (FCLL) as a classification head on top of the GPT-2 model to output sentiment class labels for input sentences. The final GPT-2 contextualized embedding of the last token serves as input to this classification head. We fine-tuned the model using two approaches: (a) training only the classification head while freezing the GPT-2 parameters, and (b) fine-tuning the entire GPT-2 model along with the classification head.

| Fine Tuning Type        | Epochs    | Batch Size | SST Acct.    | CFIMDB Acct. |
|-------------------------|-----------|------------|--------------|--------------|
| Last Linear Layer       | 10        | 8          | 0.470        | 0.865        |
| <b>Full Fine Tuning</b> | <b>10</b> | <b>8</b>   | <b>0.512</b> | <b>0.980</b> |
| Full Fine Tuning        | 10        | 16         | 0.511        | 0.976        |
| Full Fine Tuning        | 10        | 64         | 0.502        | 0.976        |
| Full Fine Tuning        | 10        | 128        | 0.504        | 0.976        |

Table 1: Performance comparison for sentiment analysis during fine-tuning

For paraphrase detection, we again added a FCLL as a classification head on top of the GPT-2 model, which takes the output hidden state representation of the last token as input and predicts a binary label. As baseline, we use full fine-tuning, where we fine-tuned the entire GPT-2 model along with the classification head. In both stage 1 (LoRA) and stage 2 (DPO) of fine-tuning, only the LoRA layers and classification head are updated. Here in stage two, DPO objective  $L_{DPO}$  is to maximize the probability of  $y_w, y_l \in \text{yes, no}$  for each input  $x$ , and the  $L_{task-specific-sft-loss}$  is the same cross-entropy loss used in stage 1 of fine-tuning.

For sonnet generation, we use the GPT-2 model directly without additional layers. We fine-tuned the model using two approaches: (a) as baseline full fine-tuning of the entire GPT-2 model, and (b) LoRA and DPO-based fine-tuning, updating only the LoRA layers. Here, also during DPO training, cross-entropy loss is used as regularization term.

## 5.4 Results

Table 1 shows fine tuning performance for sentiment analysis tasks. We can observe that model performance is relatively low when we only train the last linear layer (classification head). Further, in full fine-tuning, model performance did not show much improvement with increasing batch size, where best performance is observed with batch size of 8.

Table 2 shows supervised fine-tuning (SFT) performance for paraphrase detection. Here, higher LoRA rank consistently improves dev accuracy, with rank 32 (0.895) approaching full fine-tuning (0.898), while rank 2 trails at 0.866 — confirming that increased rank provides greater model capacity for task adaptation. Doubling training epochs from 10 to 20 yields meaningful gains across ranks (e.g., rank 32: 0.888  $\rightarrow$  0.895, rank 16: 0.885  $\rightarrow$  0.891), suggesting LoRA benefits from longer training to compensate for its constrained parameter space. Figure 1 further illustrates this training dynamics: rank 32 achieves lower training loss and higher dev accuracy than rank 16 throughout training, with both configurations showing steady improvement over 20 epochs. Notably, rank 32’s dev accuracy begins to plateau after epoch 12 while rank 16 continues to climb more gradually, suggesting that higher-rank adapters converge faster due to their greater capacity but both benefit from extended training. A learning rate of  $1e^{-4}$  proves optimal for LoRA across all ranks, roughly 10x higher than full fine-tuning’s  $1e^{-5}$ , while batch size (8 vs. 16) shows minimal impact at rank 16 — both achieving 0.885 dev accuracy — indicating LoRA is relatively robust to batch size variation at this scale. Notably, the best LoRA configuration (rank 32, 20 epochs) achieves 0.862 test accuracy, surpassing full fine-tuning’s 0.838, suggesting LoRA’s implicit regularization from low-rank constraints may improve generalization. Further, we see a 2.2x improvement in training time with LoRA. For example, full fine tuning took 6.4 hours for 10 epochs whereas LoRA fine tuning only took 2.9 hours.

Table 3 and Figure 2 present DPO results for paraphrase detection using the SFT model (LoRA rank 16,  $\alpha=32$ , dev accuracy 0.885) as the reference. We examined the impact of  $\beta$ , regularization strength  $\lambda$ , learning rate, and training epochs on DPO performance. Without regularization ( $\lambda=0.0$ ), DPO fails to improve over the SFT baseline regardless of  $\beta$  or learning rate, with all configurations plateauing at 0.884–0.885 dev accuracy. A higher  $\beta$  of 0.50 with only 5 epochs also proves insufficient, suggesting that both adequate training duration and appropriate constraint strength are needed for effective preference learning. Only with moderate regularization ( $\lambda=0.2$ ) alongside  $\beta=0.10$  and  $lr = 1e^{-5}$  does DPO surpass the baseline, reaching 0.888 dev accuracy. Figure 2 further illustrates this: regularization produces lower and more stable training losses with consistently higher dev accuracy, while the unregularized setting exhibits higher initial losses, slower convergence, and fails to exceed the reference model’s performance — confirming that regularization effectively prevents excessive policy drift during preference optimization. The best DPO configuration achieves 0.862

| SFT Type            | Epochs    | Batch Size | r         | $\alpha$  | lr          | Dev Acc.     | Test Acc.    |
|---------------------|-----------|------------|-----------|-----------|-------------|--------------|--------------|
| Full Fine-tuning    | 10        | 8          | -         | -         | 1e-5        | 0.898        | 0.838        |
| LoRA Rank 32        | 10        | 16         | 32        | 64        | 1e-4        | 0.888        | -            |
| <b>LoRA Rank 32</b> | <b>20</b> | <b>16</b>  | <b>32</b> | <b>64</b> | <b>1e-4</b> | <b>0.895</b> | <b>0.862</b> |
| LoRA Rank 16        | 10        | 8          | 16        | 32        | 1e-5        | 0.852        | -            |
| LoRA Rank 16        | 10        | 8          | 16        | 32        | 1e-4        | 0.885        | -            |
| LoRA Rank 16        | 10        | 16         | 16        | 32        | 1e-4        | 0.885        | -            |
| LoRA Rank 16        | 10        | 16         | 16        | 32        | 3e-4        | 0.883        | -            |
| LoRA Rank 16        | 20        | 16         | 16        | 32        | 1e-4        | 0.891        | -            |
| LoRA Rank 8         | 10        | 16         | 8         | 16        | 1e-4        | 0.879        | -            |
| LoRA Rank 8         | 10        | 16         | 8         | 16        | 5e-4        | 0.872        | -            |
| LoRA Rank 4         | 10        | 16         | 4         | 8         | 1e-4        | 0.875        | -            |
| LoRA Rank 2         | 10        | 16         | 2         | 4         | 1e-4        | 0.866        | -            |

Table 2: Fine Tuning Performance Comparison for paraphrase Detection as a function of learning rate, LoRA rank, batch size and training episode

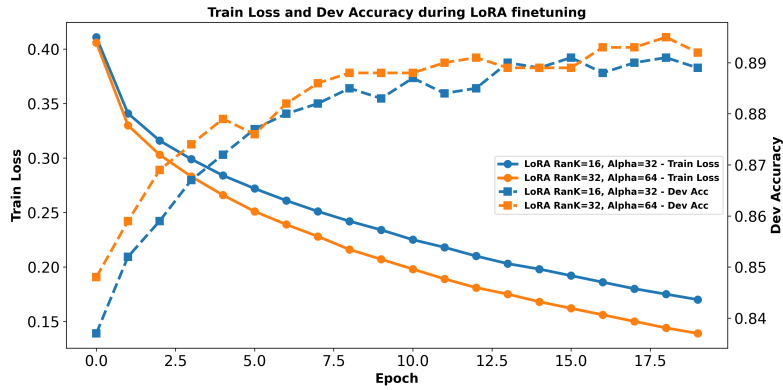


Figure 1: Training Loss and Dev accuracy as a function of training epoch for two different LoRA fine tuning at rank 16 and 32

test accuracy, matching the single-stage LoRA model (rank 32,  $\alpha=64$ ) from Table 2, suggesting that DPO with a lower-rank adapter can reach comparable generalization, though the dev-to-test gap indicates potential overfitting during the preference optimization stage. This may be due to fact that same dataset is used both during SFT and DPO stage of training.

Table4 and Table5 shows performance of supervised fine-tuning and DPO fine-tuning, respectively for sonnet generation. Instruction fine-tuning provides a modest but consistent improvement over standard full fine-tuning (42.16 vs. 42.05 chrF at 15 epochs), confirming that structured prompt formatting helps the model better learn the sonnet generation task. Adding LoRA on top of instruction tuning yields further gains, but with a clear sensitivity to rank and alpha: rank 2 with alpha=8 achieves the best SFT score (42.57), while higher ranks (4, 8) generally degrade performance, suggesting that for this creative generation task, minimal low-rank adaptation avoids overfitting and preserves the base model’s language fluency. The alpha scaling factor also matters — at rank 2, alpha=8 outperforms both alpha=16 and alpha=4, indicating a sweet spot where the LoRA update magnitude is neither too aggressive nor too conservative.

In the DPO stage table5, the best configuration ( $\beta=0.05$ ,  $cl=0.05$ ) improves chrF from the reference model’s 42.57 to 43.09, demonstrating that preference alignment meaningfully refines sonnet quality beyond supervised training alone. A moderate regularization coefficient ( $\lambda = 0.05$ ) consistently helps across  $\beta$  values, but higher regularization ( $\lambda = 0.10$ ) hurts performance by over-constraining the policy update. The  $\beta$  parameter controls the trade-off between reward maximization and staying close to the reference policy —  $\beta=0.05$  with light regularization strikes the best balance, while both lower (no regularization) and higher  $\beta$  values (0.10, 0.50) yield diminishing returns, suggesting the model benefits from gentle preference steering rather than aggressive policy shifts. However, on the test dataset, the DPO model achieved a chrF score of 42.124, slightly underperforming the reference instruction fine-tuned model (LoRA rank 2,  $\alpha=8$ ) which scored 42.264. This gap between dev and

| SFT Type                  | Epochs    | lr          | $\beta$     | $\lambda$   | Dev. Acc     | Test. Acc.   |
|---------------------------|-----------|-------------|-------------|-------------|--------------|--------------|
| Reference Model           | 10        | 1e-4        | -           | -           | 0.885        | -            |
| DPO + LoRa rank 16        | 5         | 1e-4        | 0.50        | 0.00        | 0.884        | -            |
| DPO + LoRa rank 16        | 10        | 1e-6        | 0.10        | 0.00        | 0.884        | -            |
| DPO + LoRa rank 16        | 10        | 1e-5        | 0.10        | 0.00        | 0.885        | -            |
| <b>DPO + LoRa rank 16</b> | <b>10</b> | <b>1e-5</b> | <b>0.10</b> | <b>0.20</b> | <b>0.888</b> | <b>0.862</b> |

Table 3: DPO Finetuning performance for paraphrase Detection. Here reference model is SFT fine-tuned model using LoRA rank 16 and  $\alpha$  32

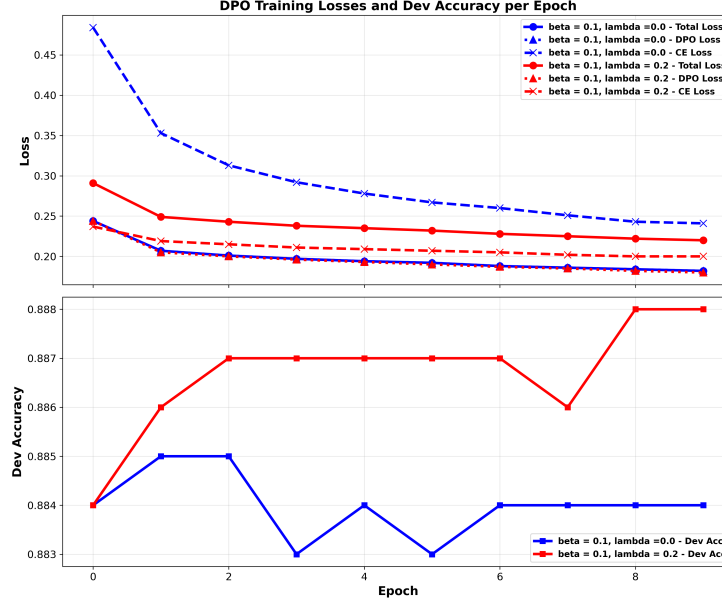


Figure 2: Top: shows total loss, dpo loss and cross-entropy loss for a fixed value of  $\beta = 0.1$  and  $cl = 0.0, cl = 0.0.2$ . Bottom: Dev accuracy as a function of training episode.

test performance suggests potential overfitting during the DPO stage, possibly exacerbated by reusing the same dataset across both supervised and preference optimization stages. Also, results in table5 are for case where losing samples are generated from reference model. The results where heuristic approach is used to generate losing samples are in appendix B, and have overall lower performance in comparison of case shown in table5.

| SFT Type                     | Epochs    | r        | $\alpha$ | lr          | Dev. chrF Score | Test. chrF Score |
|------------------------------|-----------|----------|----------|-------------|-----------------|------------------|
| Full fine-tune               | 10        | -        | -        | 1e-5        | 41.49490        | -                |
| Full fine-tune               | 15        | -        | -        | 1e-5        | 42.05032        | -                |
| Inst. fine-tune              | 15        | -        | -        | 1e-3        | 42.15986        | -                |
| Inst. fine-tune + LoRA       | 15        | 2        | 16       | 1e-3        | 42.38405        | -                |
| <b>Ins. fine-tune + LoRA</b> | <b>15</b> | <b>2</b> | <b>8</b> | <b>1e-3</b> | <b>42.56620</b> | <b>42.264</b>    |
| Inst. fine-tune + LoRA       | 15        | 2        | 4        | 1e-3        | 42.43114        | -                |
| Inst. fine-tune + LoRA       | 15        | 4        | 8        | 1e-3        | 42.16241        | -                |
| Inst. fine-tune + LoRA       | 15        | 4        | 16       | 1e-3        | 42.05092        | -                |
| Inst. fine-tune + LoRA       | 15        | 4        | 32       | 1e-3        | 42.20952        | -                |
| Inst. fine-tune + LoRA       | 15        | 8        | 16       | 1e-3        | 41.06351        | -                |
| Inst. fine-tune + LoRA       | 15        | 8        | 32       | 1e-3        | 42.04643        | -                |
| Inst. fine-tune + LoRA       | 15        | 8        | 64       | 1e-3        | 40.38437        | -                |

Table 4: Stage 1: Supervised Fine tuning performance comparison for Sonnet Generation. Here, top-p is taken as 0.80, temperature 1.2 and batch size 8 in all experiments

| SFT Type                 | Epochs   | $\beta$     | cl          | Dev. chrF Score | Test. chrF Score |
|--------------------------|----------|-------------|-------------|-----------------|------------------|
| Reference Model          | -        | -           | -           | 42.56620        | <b>42.264</b>    |
| DPO + LoRA rank 2        | 5        | 0.10        | 0.00        | 42.73782        | -                |
| DPO + LoRA rank 2        | 5        | 0.10        | 0.05        | 42.58241        | -                |
| DPO + LoRA rank 2        | 5        | 0.10        | 0.10        | 42.38341        | -                |
| DPO + LoRA rank 2        | 5        | 0.05        | 0.00        | 42.09978        | -                |
| <b>DPO + LoRA rank 2</b> | <b>5</b> | <b>0.05</b> | <b>0.05</b> | <b>43.08843</b> | <b>42.124</b>    |
| DPO + LoRA rank 2        | 5        | 0.05        | 0.10        | 42.37295        | -                |
| DPO + LoRA rank 2        | 5        | 0.50        | 0.00        | 42.30873        | -                |
| DPO + LoRA rank 2        | 5        | 0.50        | 0.05        | 42.49537        | -                |
| DPO + LoRA rank 2        | 5        | 0.50        | 0.10        | 42.24723        | -                |

Table 5: Stage 2: DPO performance for sonnet generation. Here, top-p = 0.80, temperature = 1.2 and batch size 8 in all experiments. Reference model is instruction fine-tuned using LoRA with rank 2 and  $\alpha = 8$

## 6 Analysis

Across all tasks, LoRA-based fine-tuning consistently approaches or surpasses full fine-tuning performance while significantly reducing training time, with optimal learning rates roughly an order of magnitude higher and minimal sensitivity to batch size. For sentiment analysis, full fine-tuning substantially outperforms training only the classification head, confirming that updating all model parameters is essential for effective task adaptation, while increasing batch size beyond 8 provides no meaningful benefit. The optimal LoRA rank is task-dependent — paraphrase detection benefits from higher ranks that provide greater adaptation capacity, while sonnet generation favors minimal rank to preserve the base model’s language fluency and avoid overfitting. Similarly, the alpha scaling factor requires careful tuning per task, as both overly aggressive and overly conservative update magnitudes degrade performance. Extending training epochs consistently improves LoRA performance across tasks, suggesting that the constrained parameter space benefits from longer optimization to fully converge. The two-stage DPO approach improves dev-set performance in both tasks when moderate regularization is applied, confirming that gentle preference steering prevents policy drift from the reference model while enabling meaningful gains beyond supervised fine-tuning alone. The choice of DPO hyperparameters is also task-sensitive — paraphrase detection favors higher beta with stronger regularization, while sonnet generation benefits from lower beta with lighter regularization, reflecting the different optimization landscapes of discriminative versus generative tasks. However, DPO’s test-set improvements are limited compared to its dev gains, with performance matching or slightly underperforming the single-stage instruction fine-tuning using LoRA, suggesting potential overfitting during the preference optimization stage likely exacerbated by dataset reuse across both training stages.

## 7 Conclusion

This work demonstrates that LoRA-based parameter-efficient fine-tuning is a practical and effective alternative to full fine-tuning for GPT-2 across classification and generation tasks, achieving comparable or superior performance while reducing training time by over 2x. Our experiments reveal that optimal LoRA configurations are inherently task-dependent — discriminative tasks like paraphrase detection benefit from higher ranks for greater adaptation capacity, while generative tasks like sonnet generation favor minimal ranks to preserve base model fluency, with the alpha scaling factor requiring careful calibration in both cases. The two-stage LoRA-SFT followed by DPO pipeline shows that preference optimization can yield meaningful dev-set improvements over supervised fine-tuning alone, provided that moderate regularization is applied to prevent policy drift from the reference model. However, the limited test-set gains from DPO — matching or slightly underperforming the single-stage SFT baseline — highlight a key challenge: overfitting during the preference optimization stage, likely compounded by dataset reuse across both training stages. Future work should explore using separate datasets for the SFT and DPO stages, investigating rank-adaptive LoRA strategies that dynamically adjust capacity during training, and scaling these findings to larger model architectures to assess whether the observed task-dependent trends in rank sensitivity and DPO regularization generalize beyond GPT-2.

## References

- [1] Si-Zhao Joe Qin Xiaohui Tao Fu Lee Wang Lingling Xu, Haoran Xie. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*, 2021.
- [2] Da Xiao Aaron Courville Yoshua Bengio Ian J. Goodfellow, Mehdi Mirza. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.
- [3] Phillip Wallis Zeyuan Allen-Zhu Yuanzhi Li Shean Wang Lu Wang Weizhu Chen Edward J. Hu, Yelong Shen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [4] Ari Holtzman Luke Zettlemoyer Tim Dettmers, Artidoro Pagnoni. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:1312.6211*, 2023.
- [5] Eric Mitchell Stefano Ermon Christopher D. Manning Chelsea Finn Rafael Rafailov, Archit Sharma. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.
- [6] Stanislaw Jastrzebski Bruna Morrone Quentin de Laroussilhe Andrea Gesmundo Mona Attariyan Sylvain Gelly Neil Houlsby, Andrei Giurgiu. Parameter-efficient transfer learning for nlp. *arXiv preprint arXiv:1902.00751*, 2019.
- [7] Percy Liang Xiang Lisa Li. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2019.
- [8] Noah Constant Brian Lester, Rami Al-Rfou. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.
- [9] Yoav Goldberg Elad Ben-Zaken, Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.
- [10] Zheng Wang Atticus Geiger Dan Jurafsky Christopher D. Manning Christopher Potts Zhengxuan Wu, Aryaman Arora. Reft: Representation finetuning for language models. *arXiv preprint arXiv:2404.03592*, 2025.
- [11] Steve Nwaiwu. Parameter-efficient fine-tuning for low-resource text classification: a comparative study of lora, ia3, and reft. *Frontiers in big data*, 8, 1677331 doi:10.3389/fdata.2025.1677331, 2025.
- [12] Nathan Lambert. Reinforcement learning from human feedback. *arXiv preprint arXiv:2504.12501*, 2025.
- [13] Leilei Gan Shuai Zhao Zongrui Li Ruirui Lei Wanggui He Luu Anh Tuan Long Chen Hao Jiang Zhou Zhao Fei Wu Wenyi Xiao, Zechuan Wang. A comprehensive survey of direct preference optimization: Datasets, theories, variants, and applications. *arXiv preprint arXiv:2410.15595*, 2024.
- [14] Bilal Piot Daniel Guo Daniele Calandriello Michal Valko Rémi Munos Mohammad Gheshlaghi Azar, Mark Rowland. A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036*, 2023.
- [15] Niklas Muennighoff Dan Jurafsky Douwe Kiela Kawin Ethayarajh, Winnie Xu. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

## A Instruction Prompts for Sonnet Generation Tasks

For instruction fine-tuning of sonnet generation, we considered two tasks which are (a) model needs to generate 14 lines of Shakespearean style sonnet and (b) model is given 3 lines of sonnet and it needs to complete the remaining 11 lines of sonnet. The instruction prompt for both cases is given below.

Training instruction Prompts where the model need to generate Shakespearean style sonnet.

```
{
  "instruction": "Write a Shakespearean sonnet. A Shakespearean sonnet
                consists of 14 lines written in iambic pentameter
                (10 syllables per line,alternating unstressed and stressed
                syllables). It is structured as three quatrains followed by a
                closing couplet: lines 1-4 form the first quatrain (ABAB),
                lines 5-8 the second quatrain (CDCD),lines 9-12 the third
                quatrain (EFEF), and lines 13-14 form the closing couplet
                (GG). Within each quatrain, the 1st and 3rd lines rhyme with
                each other (A, C, or E), and the 2nd and 4th lines rhyme
                with each other (B, D, or F). The final two lines form
                a rhyming couplet (GG).",
  "input": "",
  "response": sonnet_text
}
```

Training instruction Prompts where the model need to complete Shakespearean style sonnet, where first three lines of sonnet is given as input and model to complete the remaining 11 lines of sonnet.

```
{
  "instruction": "Complete this Shakespearean sonnet.
                You are given the first 3 lines and need to complete the remaining
                11 lines. A Shakespearean sonnet consists of 14 lines written
                in iambic pentameter (10 syllables per line, alternating unstressed
                and stressed syllables). It is structured as three quatrains
                followed by a closing couplet: lines 1-4 form the first quatrain
                (ABAB),lines 5-8 the second quatrain (CDCD), lines 9-12 the third
                quatrain (EFEF), and lines 13-14 form the closing couplet (GG).
                Within each quatrain, the 1st and 3rd lines rhyme with each other
                (A, C, or E), and the 2nd and 4th lines rhyme with each other
                (B, D, or F). The A rhyme is already demonstrated by the given
                lines 1 and 3 - your next line (line 4) must rhyme with line 2 (B)
                to complete the first quatrain. The final two lines form a
                rhyming couplet (GG).",
  "input": "First 3 lines of sonnet,",
  "response": sonnet_text
}
```

## B Appendix DPO performance for sonnet generation where loosing samples are constructed using heuristic approach

Table 6 shows the DPO training performance for sonnet generation where loosing preference pairs data is generated using heuristic approach

| SFT Type          | Epochs | $\beta$ | cl   | Dev. CHRF Score | Test. CHRF Score |
|-------------------|--------|---------|------|-----------------|------------------|
| Reference Model   | -      | -       | -    | 42.56620        | <b>42.264</b>    |
| DPO + LoRA rank 2 | 5      | 0.10    | 0.00 | 42.67888        | -                |
| DPO + LoRA rank 2 | 5      | 0.10    | 0.05 | 42.94622        | -                |
| DPO + LoRA rank 2 | 5      | 0.10    | 0.10 | 42.64702        | -                |
| DPO + LoRA rank 2 | 5      | 0.05    | 0.05 | 42.36069        | -                |

Table 6: Stage 2: DPO performance for sonnet generation. Here, top-p = 0.80, temperature = 1.2 and batch size 8 in all experiments. Reference model is instruction fine-tuned using LoRA with rank 2 and  $\alpha = 8$