

Full Fine-Tuning vs. LoRA: Parameter-Efficient Adaptation of GPT-2 for Paraphrase Detection and Sonnet Generation

Stanford CS224N Default Project

Megha Bindiganavale
Department of Computer Science
Stanford University
mbindiga@stanford.edu

Rydhm Goyal
Department of Computer Science
Stanford University
rydham@stanford.edu

Abstract

Fine-tuning all the parameters of a large language model works well but is expensive. LoRA (Low-Rank Adaptation) offers a cheaper alternative by only training small low-rank matrices while keeping the rest of the model frozen. We study how LoRA compares to full fine-tuning on GPT-2 (124M parameters) on two tasks: paraphrase detection (a classification task) and sonnet generation (an open-ended generation task). We implement LoRA from scratch, adding trainable low-rank matrices to the query and value projections in each attention layer. For paraphrase detection, LoRA with just 147K trainable parameters (0.12% of the model) reaches 85.9% dev accuracy versus 88.7% for full fine-tuning, a gap of only 2.8 points. For sonnet generation, however, LoRA performs much worse: its outputs are often incoherent, scoring 17.7 chrF compared to 25.8 in our initial full fine-tuning setup, and increasing the rank from 1 to 16 does not materially improve generation quality. We further study decoding and preference-based post-training for sonnet generation. Nucleus sampling substantially outperforms beam search, and Direct Preference Optimization (DPO) provides an additional improvement on top of supervised fine-tuning, raising dev chrF from 41.5 to 42.0. Our error analysis shows that LoRA’s paraphrase mistakes concentrate on high-overlap sentence pairs, while its sonnet failures stem from an inability to shift the output distribution enough toward Shakespearean language.

1 Key Information

- Mentor: Shicheng Liu
- External Collaborators: None
- Sharing project: No

2 Introduction

GPT-2 [1] and other large pretrained language models can be fine-tuned to perform well on many downstream NLP tasks. The problem is that full fine-tuning means updating all 124M parameters, which takes a lot of compute and memory. Parameter-efficient fine-tuning (PEFT) methods try to get around this by only updating a small number of parameters while freezing the rest.

LoRA [2] is a popular PEFT method that adds small trainable low-rank matrices to selected layers of a frozen pretrained model. The idea is that the weight changes needed for a new task are low-rank, so you can get away with much fewer trainable parameters. LoRA showed strong results on NLU benchmarks in the original paper, but it is less clear how well it works for open-ended generation, especially with smaller models and limited training data.

We compare full fine-tuning against LoRA on two tasks using GPT-2 base: (1) **paraphrase detection** on Quora Question Pairs, which is a binary classification task, and (2) **sonnet generation**, where the model has to complete Shakespeare sonnets given the first three lines. These two tasks test very different things: classification just needs the model to pick between two options, while generation needs it to produce coherent text token by token.

We implement LoRA from scratch, targeting the query and value projections in each attention layer. We try different ranks ($r \in \{1, 2, 4, 16\}$) and do error analysis to compare where LoRA and full fine-tuning make different mistakes. Our main finding is that LoRA does surprisingly well on classification (85.9% vs. 88.7%) but basically fails on generation (chrF 17.7 vs. 25.8), even with higher ranks. This lines up with recent work showing LoRA struggles on generation tasks [3, 4].

3 Related Work

LoRA. Hu et al. [2] proposed Low-Rank Adaptation, which freezes pretrained weights and adds trainable rank decomposition matrices $\Delta W = BA$ to selected layers. They found that targeting query and value projections (W_q, W_v) was optimal for a fixed parameter budget and demonstrated competitive performance on GPT-3 175B for NLU tasks.

LoRA limitations. Biderman et al. [3] found that LoRA substantially underperforms full fine-tuning on code generation, requiring $16\times$ more training tokens to match performance. They also showed that LoRA preserves more of the base model’s diversity, which is beneficial for some tasks but harmful when the target distribution differs significantly from pretraining. Kotha et al. [4] showed that full fine-tuning learns weight perturbations with $10\text{--}100\times$ higher rank than typical LoRA configurations, and that the two methods produce structurally different weight updates. Classification tasks are considerably easier than generation tasks for LoRA, which our experiments confirm.

Preference optimization. Large language models are often aligned with human preferences using Reinforcement Learning from Human Feedback (RLHF). Ouyang et al. [5] demonstrated the effectiveness of this approach by training a reward model from human preference data and then optimizing the policy with Proximal Policy Optimization (PPO). However, RLHF is relatively complex and computationally expensive because it requires a separate reward model and online policy optimization. Rafailov et al. [6] proposed Direct Preference Optimization (DPO), which avoids explicit reward-model training by directly optimizing the policy on pairwise preference data relative to a frozen reference model. DPO is substantially simpler to implement than standard RLHF while achieving competitive results. In this project, we use DPO as a post-training refinement step for both paraphrase detection and sonnet generation.

GPT-2. Radford et al. [1] introduced GPT-2, a transformer-based autoregressive language model. We use GPT-2 base (12 layers, 768 hidden dimensions, 12 attention heads, 124M parameters) as our base model for both tasks.

Attention mechanism. Our implementation builds on the transformer architecture of Vaswani et al. [7], using multi-head causal self-attention with scaled dot-product attention.

4 Approach

4.1 Baseline: Full Fine-Tuning

We implemented GPT-2 from scratch, including multi-head causal self-attention, 12 transformer layers (attention + feed-forward with GELU activation), and the AdamW optimizer with bias correction and decoupled weight decay.

Paraphrase detection. We use a cloze-style approach. The input is formatted as: "Question 1: [s1] Question 2: [s2] Are these questions asking the same thing?" We extract the last token’s hidden state h_{last} and compute logits by dotting against the pretrained embedding vectors for “no” (token 3919) and “yes” (token 8505):

$$\text{logits} = h_{\text{last}} \cdot E_{[\text{no}, \text{yes}]}^{\top} \quad (1)$$

This reuses GPT-2’s pretrained vocabulary knowledge rather than adding a new classification head. We train with cross-entropy loss over the two logits.

Sonnet generation. For sonnet generation, we train GPT-2 as an autoregressive language model over the full sonnet text. Given a tokenized sonnet, the model returns a hidden state at every position, which we project to vocabulary logits using the pretrained output embedding. Unlike paraphrase detection, which uses only the final hidden state, sonnet generation requires predicting the next token at every position. We therefore train with teacher forcing and minimize cross-entropy over the full sequence, using the token at position $t + 1$ as the target for the logits at position t .

At inference time, generation is autoregressive: we provide the first three lines of a held-out sonnet as the prompt, sample the next token from the predicted distribution, append it to the context, and repeat until end-of-text or a length limit is reached. We experimented with both nucleus sampling and beam search. In our final system, nucleus sampling worked best, so we use temperature $\tau = 1.0$ and top- $p = 0.9$.

4.2 Extension 1: LoRA

We implement LoRA from scratch following Hu et al. [2]. For each target linear layer with frozen weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we add trainable low-rank matrices $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}} \times r}$, where $r \ll \min(d_{\text{in}}, d_{\text{out}})$ is the rank. The forward pass becomes:

$$y = Wx + b + \frac{\alpha}{r}(xA^\top B^\top) \quad (2)$$

where α is a scaling hyperparameter. Following the original paper, A is initialized with small random values and B is initialized to zeros, so the LoRA contribution is zero at the start of training and the model begins from the exact pretrained behavior.

We target the query (W_q) and value (W_v) projections in all 12 attention layers, as Hu et al. found this to be optimal. All other parameters are frozen so only the LoRA matrices are trained.

For GPT-2 base with $d = 768$, each LoRA pair adds $2 \times r \times 768$ parameters per layer. With 12 layers and 2 projections per layer, the total trainable parameters for rank r are:

$$\text{trainable params} = 12 \times 2 \times 2 \times r \times 768 = 36,864 \times r \quad (3)$$

4.3 Extension 2: Direct Preference Optimization

As a second extension, we applied Direct Preference Optimization (DPO) [6]. DPO trains a policy model π_θ to prefer better responses over worse ones relative to a frozen reference model π_{ref} . Given an input x , a preferred response y_w , and a dispreferred response y_l , the DPO objective is

$$\mathcal{L}_{\text{DPO}}(\theta) = -\log \sigma \left(\beta [\log \pi_\theta(y_w | x) - \log \pi_\theta(y_l | x) - \log \pi_{\text{ref}}(y_w | x) + \log \pi_{\text{ref}}(y_l | x)] \right), \quad (4)$$

where σ is the logistic sigmoid and β controls the strength of the preference signal.

For paraphrase detection, we apply DPO to the cloze-style yes/no logits. The gold label is treated as the preferred response y_w , while the opposite label is treated as the dispreferred response y_l . Since the output space is binary, DPO in this setting reduces to encouraging the model to score the correct label more highly than the incorrect one relative to the frozen reference model.

For sonnet generation, we construct preference triples (x, y_w, y_l) where x is the first three lines of a sonnet, y_w is the true continuation, and y_l is the continuation from another sonnet. Unlike paraphrase detection, the responses are full sequences rather than single labels, so we compute sequence-level log-probabilities by summing token log-probabilities over the completion:

$$\log \pi_\theta(y | x) = \sum_{t=1}^{|y|} \log \pi_\theta(y_t | x, y_{<t}). \quad (5)$$

We then apply the same DPO objective using these sequence-level log-probabilities. In the sonnet setting, we found that DPO works best as a refinement step on top of a supervised fine-tuned model rather than when trained directly from pretrained GPT-2.

5 Experiments

5.1 Data

For paraphrase detection, we use the Quora Question Pairs dataset with 283K training pairs and 40K dev pairs. Each example contains two questions and a binary label indicating whether they are paraphrases. We preprocess following the provided starter code.

For sonnet generation, we use a collection of 144 Shakespeare sonnets for training and 12 held-out sonnets (145–156) for evaluation. The held-out sonnets provide only the first 3 lines; the model must generate the remaining lines.

5.2 Evaluation

For paraphrase detection, we report dev set accuracy (percentage of correctly classified pairs).

For sonnet generation, we report training loss (cross-entropy) and chrF, a character n-gram F-score computed against the ground-truth Shakespeare completions. We also perform qualitative analysis of generated text, examining vocabulary usage, line structure, and coherence.

5.3 Experimental Details

All experiments use GPT-2 base (124M parameters) on a single NVIDIA T4 GPU (15GB VRAM).

Full fine-tuning. For paraphrase detection: 3 epochs, batch size 8, learning rate 1×10^{-5} , AdamW with no weight decay, max sequence length 128 tokens. For sonnet generation, full fine-tuning uses 10 epochs, batch size 8, learning rate 1×10^{-5} , AdamW with no weight decay, and maximum sequence length 256. For decoding, the final full fine-tuning model uses temperature $\tau = 1.0$ and nucleus sampling with $\text{top-}p = 0.9$.

LoRA. For paraphrase detection: 3 epochs, batch size 8, learning rate 1×10^{-4} , $\alpha = 8$, AdamW with no weight decay, max sequence length 128 tokens. We found that LoRA requires a higher learning rate than full fine-tuning (1×10^{-4} vs. 1×10^{-5}) since only a small number of parameters are updated. For sonnet generation: 3 epochs, batch size 8, learning rate 1×10^{-4} , $\alpha = 8$.

We sweep over ranks $r \in \{1, 2, 4\}$ for both tasks and additionally test $r = 16$ for sonnet generation.

DPO. For paraphrase detection, DPO uses 10 epochs, batch size 16, learning rate 1×10^{-5} , and $\beta = 0.1$. For sonnet generation, we tried two settings. Training DPO directly from pretrained GPT-2 used 3 epochs, batch size 1, learning rate 1×10^{-5} , and $\beta = 0.1$. Our final sonnet DPO setup initialized both the policy and frozen reference model from the 10-epoch full fine-tuning checkpoint, then trained for 5 epochs with batch size 2, learning rate 1×10^{-5} , and $\beta = 0.15$.

5.4 Results

5.4.1 Paraphrase Detection

Config	Trainable Params	Epoch 0	Epoch 1	Epoch 2
Full fine-tuning	124M (100%)	0.866	0.878	0.887
LoRA $r = 4$	147K (0.12%)	0.827	0.848	0.859
LoRA $r = 2$	74K (0.06%)	0.835	0.850	0.856
LoRA $r = 1$	37K (0.03%)	0.833	0.844	0.853
Staff Baseline	–	–	–	0.882

Table 1: Paraphrase detection dev accuracy on Quora Question Pairs.

Full fine-tuning achieves 88.7% dev accuracy, exceeding the staff baseline of 88.2%. LoRA achieves 85.3–85.9% depending on rank, a gap of only 2.8–3.4 percentage points despite using less than 0.12% of the parameters. Interestingly, all three LoRA ranks perform similarly, suggesting that even rank 1 captures enough of the task-relevant weight update for this binary classification problem.

We also applied DPO to paraphrase detection as an additional extension. Starting from the same GPT-2 base model, DPO improved dev accuracy to 89.3% with dev F1 0.886 after 10 epochs, slightly above the full fine-tuning baseline. Since paraphrase detection only requires the model to separate two output tokens (“yes” and “no”), preference optimization is relatively stable in this setting and gives a small but measurable gain.

5.4.2 Sonnet Generation

Config	Trainable Params	Epoch 0	Epoch 1	Epoch 2
LoRA $r = 1$	37K (0.03%)	5.195	5.117	4.975
LoRA $r = 2$	74K (0.06%)	5.191	5.127	4.964
LoRA $r = 4$	147K (0.12%)	5.202	5.141	4.994
LoRA $r = 16$	590K (0.47%)	5.205	5.129	5.001

Table 2: Sonnet generation training loss (cross-entropy) across LoRA ranks.

Config	Corpus chrF
Full fine-tuning	25.8
LoRA $r = 4$	17.7
LoRA $r = 16$	14.2

Table 3: Sonnet generation chrF scores against ground-truth Shakespeare completions. Increasing rank to 16 actually hurts, possibly due to overfitting on the small dataset of 144 sonnets.

All LoRA ranks converge to similar training loss (~ 5.0) after 3 epochs, and increasing rank from 1 to 16 provides no meaningful improvement. The chrF score tells a clearer story: full fine-tuning scores 25.8 while LoRA scores 17.7, a 31% reduction. (The results in Table 4 below use our final tuned decoding setup, whereas Table 3 reports the earlier full fine-tuning vs. LoRA comparison under the initial generation configuration.)

Config	Dev chrF
Full fine-tuning (tuned top- p sampling)	41.507
DPO from pretrained GPT-2	41.732
Full fine-tuning + DPO	42.009

Table 4: Final sonnet generation results after decoding tuning and DPO.

Decoding strategy had a large effect on sonnet quality. We tried top- k sampling, nucleus sampling, and beam search. Among these, nucleus sampling worked best: with temperature $\tau = 1.0$ and top- $p = 0.9$, the full fine-tuning model reached 41.507 chrF on the dev set, slightly above the staff baseline of 41.414. In contrast, beam search consistently underperformed, giving chrF in the high-20s (roughly 27–30 depending on beam width and repetition penalty). Qualitatively, beam search tended to produce shorter and more repetitive completions, while sampling preserved more lexical diversity and poetic flow.

We then applied DPO as a post-training refinement for sonnet generation. Training DPO directly from pretrained GPT-2 improved chrF to 41.732, already competitive with the tuned full fine-tuning baseline. However, the best setup was to start from the 10-epoch supervised fine-tuning checkpoint and then run DPO for 5 additional epochs with $\beta = 0.15$. In this setting, the DPO loss decreased from 1.697 to 0.327 over 5 epochs, and dev chrF improved further to 42.009. This made DPO the best-performing sonnet generation method we tried.

6 Analysis

6.1 Paraphrase Detection Error Analysis

We compare predictions from full fine-tuning and LoRA ($r = 4$) on the 40,429 dev examples to understand where LoRA fails.

Category	Count
Both correct	33,408
Both wrong	3,346
Full FT correct, LoRA wrong	2,470
LoRA correct, Full FT wrong	1,205

Table 5: Agreement between full fine-tuning and LoRA on dev set.

Of the 2,470 examples LoRA gets wrong that full fine-tuning gets right, 1,320 are false negatives (LoRA misses true paraphrases) and 1,150 are false positives (LoRA incorrectly marks non-paraphrases as paraphrases).

LoRA errors concentrate on high-overlap pairs. The average word overlap for examples LoRA gets wrong is 0.617, compared to 0.491 for examples it gets right. This suggests LoRA struggles with sentence pairs that share many words but differ in meaning, which are exactly the cases that require deeper semantic understanding. For example, LoRA incorrectly classifies “What is the first computer?” vs. “What was your first computer and when?” as paraphrases: the high lexical overlap masks a meaningful difference.

LoRA errors are on shorter sentences. The average combined word count for LoRA errors is 19.7, versus 22.5 for correct predictions. Shorter questions may have more ambiguous phrasing, making fine-grained semantic distinctions harder with limited model capacity.

6.2 Sonnet Generation Quality Analysis

Metric	Full FT	LoRA
Unique words	698	785
Words in Shakespeare vocab	62.0%	40.9%
Modern/web words	0	9
Avg words per line	8.6	14.9
Corpus chrF	25.8	17.7

Table 6: Sonnet generation quality comparison.

The difference in quality between full fine-tuning and LoRA is pretty clear. Full fine-tuning generates text that mostly stays within Shakespearean vocabulary (62% of unique words appear in the training sonnets) and maintains line structure close to iambic pentameter (8.6 words per line, compared to the target of ~ 10). LoRA generations, by contrast, use only 40.9% Shakespearean vocabulary and produce prose-like blocks averaging 14.9 words per line. LoRA outputs contain 9 modern/web words (e.g., “university,” “instagram,” “amazon”) while full fine-tuning produces none.

Qualitatively, full fine-tuning generates recognizably poetic text (e.g., “To another say crease where no man could make it freeze”), while LoRA produces incoherent text with random names, numbers, and modern references (e.g., “Gabriel Agabal summoned Evil over her, Enwise Mistress of Lunya”).

Increasing rank does not help. We tested rank 16 (590K parameters, 0.47%) in addition to ranks 1, 2, and 4. All ranks converge to nearly identical training loss (~ 5.0) and produce similarly incoherent text. This suggests that the problem is not about having too few parameters, but rather that low-rank updates to just the attention projections are not enough to change the output distribution over 50,257 vocabulary tokens.

6.3 Effect of DPO on Sonnet Generation

Direct Preference Optimization gave the best final sonnet generation results. Starting from the 10-epoch supervised fine-tuned checkpoint and training on preference pairs of the form (x, y_w, y_l) , where x is the first three lines of a sonnet, y_w is the true continuation, and y_l is the continuation from another sonnet, DPO improved dev chrF from 41.5 to 42.0. The improvement is not dramatic, but it is consistent and shows that preference optimization can refine an already competent generative model.

The comparison between “DPO from scratch” and “DPO from supervised fine-tuning” is also informative. When trained directly from pretrained GPT-2, DPO reached 41.7 chrF, which is competitive but still below the 42.0 obtained when starting from the supervised fine-tuned checkpoint. This suggests that DPO is most useful as a second-stage refinement procedure rather than as a replacement for next-token training. In other words, supervised fine-tuning teaches the model the basic Shakespearean distribution, while DPO provides a smaller correction that prefers better continuations over weaker ones.

Qualitatively, the DPO outputs remain closer to the target continuation distribution and are somewhat less likely to drift into generic or loosely related continuations. This is consistent with the DPO objective: instead of only maximizing likelihood, it explicitly increases the relative probability of preferred completions over dispreferred alternatives. For open-ended generation, this kind of preference shaping appears to be a better final adjustment than simply continuing standard language-model training.

6.4 Why LoRA Fails on Generation but Not Classification

We think this difference between tasks comes down to what the model actually needs to learn, which is also supported by Biderman et al. [3] and Kotha et al. [4]:

- **Classification** only needs the model to tell apart two tokens (“yes” and “no”). A small low-rank tweak to the hidden states is enough for this.
- **Generation** needs the model to change the probability distribution over all 50,257 tokens in the vocabulary at every position. This is a much bigger change than what low-rank updates to just the attention projections can handle.

Kotha et al. showed that full fine-tuning produces weight perturbations with 10–100× higher rank than typical LoRA configurations, explaining why the gap is especially large for generation tasks.

7 Conclusion

We compared full fine-tuning, LoRA, and Direct Preference Optimization (DPO) on GPT-2 base for paraphrase detection and sonnet generation. Our main findings are:

1. LoRA achieves 85.9% accuracy on paraphrase detection with only 0.12% trainable parameters, closing to within 2.8 points of full fine-tuning (88.7%).
2. LoRA fails on sonnet generation regardless of rank ($r \in \{1, 2, 4, 16\}$), producing incoherent text with a chrF of 17.7 versus much stronger full fine-tuning results.
3. Decoding strategy matters substantially for sonnet generation: nucleus sampling performs far better than beam search, and with tuned top- p sampling full fine-tuning reaches 41.5 chrF on the dev set.
4. DPO improves both tasks as a post-training refinement. On paraphrase detection it raises dev accuracy to 89.3%, and on sonnet generation it improves chrF from 41.5 for supervised fine-tuning alone to 42.0 when applied on top of the full fine-tuning checkpoint.
5. Error analysis shows LoRA’s paraphrase errors concentrate on high-overlap sentence pairs requiring deeper semantic understanding, while generation failures reflect an inability to shift the output distribution toward Shakespearean language.

Overall, LoRA works well when the task only requires a small change to the model’s behavior (like classification), but it is not enough when the model must learn a very different output distribution (like generating Shakespearean sonnets). For generation, full fine-tuning remains much stronger, and DPO provides a useful second-stage refinement on top of it. In future work, it would be interesting to apply LoRA to more layers (keys, output projections, feed-forward layers), use a larger base model like GPT-2 Large where LoRA has shown better generation results [3], or combine LoRA with preference optimization methods.

Team Contributions

Megha Bindiganavale: Implemented GPT-2 base model (multi-head causal self-attention, transformer layers), implemented paraphrase detection (cloze-style forward pass) and ran full fine-tuning for it, implemented LoRA from scratch, integrated LoRA into both paraphrase detection and sonnet generation, ran all LoRA rank sweep experiments, performed error analysis, and co-wrote the report.

Rydhm Goyal: Implemented the AdamW optimizer, the sonnet generation pipeline (full-sequence forward pass, autoregressive decoding, nucleus sampling, and beam search), ran full fine-tuning experiments for sentiment analysis and sonnet generation, implemented DPO for both paraphrase detection and sonnet generation, ran the DPO experiments and decoding comparisons, and co-wrote the report.

References

- [1] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.
- [2] 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.
- [3] Dan Biderman, Jose Gonzalez Ortiz, Jacob Portes, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P Cunningham. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*, 2024.
- [4] Reena Kotha, Shruti Mishra, and Srinadh Bhojanapalli. Lora vs full fine-tuning: An illusion of equivalence. *arXiv preprint arXiv:2410.21228*, 2024.
- [5] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, 2022.
- [6] 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*, volume 36, 2023.
- [7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.