

Optimizing GPT-2 for Downstream Tasks: An Exploration of PEFT, Preference Optimization, and SMART

Stanford CS224N Default Project

Anastasiya Masalava Michael Rybalkin Eva Casto

Department of Computer Science

Stanford University

{masalava, rybalkin, ecasto}@stanford.edu

Abstract

Adapting large pre-trained models to diverse downstream tasks requires balancing computational efficiency, stylistic alignment, and model robustness. In this project, we adapt GPT-2 to three downstream tasks: paraphrase detection, sonnet generation, and sentiment analysis. Despite GPT-2's strong language modeling capabilities, naive fine-tuning often fails to preserve output quality while remaining parameter-efficient and robust to adversarial inputs. We propose and evaluate three extensions: Parameter-Efficient Fine-Tuning (PEFT) with LoRA, Direct Preference Optimization (DPO), and Smoothness-Inducing Adversarial Regularization (SIAR). On the test set, DPO (GPT-2 Medium) achieved the highest performance with 0.899 accuracy and 42.636 chrF on paraphrase and sonnet tasks, respectively. LoRA (GPT-2 Large) reached 0.894 accuracy on paraphrase detection while updating only 3.31% of parameters, while SIAR achieved 0.895 accuracy and reduced model brittleness by $2\times$ via a halved adversarial flip rate.

1 Key Information

Mentor: David Anugraha **Collaborators:** None **Sharing:** No **Late days:** (pooled/size/applied): 9/3/3 (*Anastasiya Masalava: 6, Michael Rybalkin: 0, Eva Casto: 4*)

2 Introduction

Recent advances in Natural Language Processing have been driven by large language models (LLMs), operating through unsupervised pre-training on massive text corpora, followed by task-specific fine-tuning [1]. Generative Pretrained Transformer (GPT-2) models demonstrate that a decoder-only Transformer can effectively model text by predicting the next word based only on leftward context, opening up new possibilities in text generation and model quality across a wide range of NLP downstream tasks such as question answering and sentiment analysis.

However, adapting large pre-trained language models to specialized tasks remains challenging, providing the motivation for this project. Fine-tuning large models can be computationally expensive and prone to overfitting when training data is limited. We implement a base GPT-2 model and investigate methods for improving its performance on two downstream tasks: paraphrase detection on the Quora dataset and Shakespearean sonnet generation. Paraphrase detection measures whether the model can capture semantic equivalence across sentence pairs, while sonnet generation evaluates its ability to follow strict poetic structure while producing fluent text. To address the challenges of efficient adaptation and model alignment, we explore three extensions: Low-Rank Adaptation (LoRA) for parameter-efficient fine-tuning, Direct Preference Optimization (DPO) for preference alignment, and Smoothness-Inducing Adversarial Regularization (SIAR) for improved robustness.

3 Related Work

The paradigm of pre-training followed by task-specific fine-tuning has established GPT-style architectures as a foundation for diverse downstream tasks [2]. As models scale, the research focus has shifted from simple adaptation to addressing the inherent challenges of computational cost, alignment with complex constraints, and representation robustness. The following work situates our study within research on efficient adaptation, semantic similarity modeling, stylistic alignment in generation, and robustness of transformer representations.

Efficiency and Adaptation While full fine-tuning is the performance gold standard, its memory requirements often exceed hardware constraints, leading to Parameter-Efficient Fine-Tuning (PEFT) [3]. Foundational methods like Adapters [4] and Prefix Tuning [5] localize task knowledge in small parameter subsets. Low-Rank Adaptation (LoRA) [6] further optimizes this by decomposing weight updates into low-rank matrices. We evaluate whether LoRA enables scaling GPT-2 to larger variants for semantic tasks such as paraphrase detection.

Semantic Similarity and Paraphrase Detection Recognizing equivalent meanings across varying syntactic structures is a core challenge in NLP [7]. Early approaches focused on traditional word embeddings and alignment, but transformer-based models like BERT [8] and Sentence-BERT [9] significantly influenced the field by producing semantically meaningful embeddings that capture context-dependent nuances. In this work, we examine whether GPT-2 can perform paraphrase classification effectively while maintaining parameter efficiency and robustness.

Stylistic Alignment Poetic generation requires balancing fluency with rigid structural rules. Early RNN and LSTM approaches struggled with long-range structural consistency [10]. While supervised fine-tuning (SFT) is now standard, it often overfits on small datasets like Shakespearean sonnets. To improve qualitative alignment, Direct Preference Optimization (DPO) [11] provides a supervised alternative to Reinforcement Learning from Human Feedback (RLHF) [12]. We investigate if DPO can steer models toward formal constraints without the overhead of traditional RL.

Robustness and Generalization Transformers often exhibit "brittleness", where minor embedding perturbations cause significant output shifts [13, 14]. The SMART framework [15] addresses this using adversarial regularization to encourage output consistency. We explore whether such smoothness-inducing techniques can stabilize both classification boundaries and generative stylistic patterns simultaneously.

4 Approach

4.1 Baseline Implementation

Our baseline implementation utilizes the foundational GPT-2 architecture provided in the default project instructions [16]. The baseline model implements standard transformer components, including a sequence of Layer Normalization, Masked Multi-Head Self-Attention, and a GELU-activated MLP, all bridged by residual connections. We used the baseline GPT-2 architecture to address our three downstream tasks.

For the *sentiment analysis* task, our baseline `GPT2SentimentClassifier` extracts the hidden state of the final token for each input sequence, applies a dropout layer, and passes it through a linear classifier, outputting logits corresponding to the sentiment classes. For the *paraphrase detection* task, `ParaphraseGPT` treats the problem as a cloze-style generation task by projecting the last token’s hidden state through a 2D linear layer mapped specifically to the byte-pair encoding indices for the words "yes" and "no". Lastly, for *sonnet generation*, our `SonnetGPT` performs standard autoregressive language modeling by outputting logits for every token in the sequence and employing top-p (nucleus) sampling with temperature scaling to iteratively generate the next token.

4.2 Parameter-Efficient Fine-Tuning with LoRA

LoRA hypothesizes that the change in model weights during fine-tuning has a low intrinsic rank. Instead of updating the full pre-trained weight matrix $W \in \mathbb{R}^{d_{out} \times d_{in}}$, LoRA freezes W and injects

trainable rank decomposition matrices. Let $\Delta W = BA$, where $B \in \mathbb{R}^{d_{out} \times r}$ and $A \in \mathbb{R}^{r \times d_{in}}$, and the rank $r \ll \min(d_{in}, d_{out})$. During the forward pass, for an input x , the modified output is computed as:

$$h = Wx + \frac{\alpha}{r}BAx$$

where α is a constant scaling factor.

In our implementation, we created a custom `LoRALinear` module to wrap the standard `PyTorch nn.Linear` layers. We freeze the original weights and biases and initialize the low-rank matrices A and B . To ensure the initial adaptation is zero, A uses a Kaiming uniform initialization and B is initialized with zeros. For an input tensor x , our forward pass computes the standard linear transformation and adds the scaled low-rank update:

$$\text{output} = \text{base_out} + (xA^TB^T)\frac{\alpha}{r}$$

We applied our `LoRALinear` wrapper across the GPT-2 transformer blocks by targeting the attention matrices (query, key, value, attention_dense) as well as the MLP layers (interm_dense, out_dense). For the *paraphrase detection task*, we integrated LoRA into our `ParaphraseGPT` architecture. By freezing the base GPT-2 parameters, we only computed gradients for the injected low-rank adapters and the final 2D linear classification head. This allowed us to experiment with GPT-2 Medium and GPT-2 Large within our GPU memory constraints. For the *sonnet generation task*, we applied LoRA to our `SonnetGPT` model. Because the training dataset consists of only 143 sonnets, full-model fine-tuning is very likely to overfit. By applying LoRA with a low rank and weight decay, we prevented the model from memorizing the training data, forcing it to rely on the general language understanding it acquired during pre-training while adapting to the Shakespearean domain.

4.3 Preference Optimization with Pair-wise Data

Direct Preference Optimization is a method of aligning language models with human preferences using pairwise data. The paired examples used in DPO contain a winning (y_w) and losing (y_l) output from a prompt x . The model optimizes the following objective, learning to give higher probability to preferred outputs.

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

DPO contains a temperature parameter β , as well as a frozen reference model (π_{ref}) and the model being trained (π_{θ}).

While DPO is primarily suited for generation tasks, we successfully applied it to the binary classification task of *paraphrase detection*. Since the dataset already provides high-quality binary labels, we constructed pairwise preference examples directly in the answer space. To do this, the preferred output is the correct answer ("yes" to paraphrases and "no" otherwise), and the non-preferred output is the opposite token. This yields a pairwise comparison for every training instance without changing the input sentences.

As stated, DPO performs better with generation tasks like that of the *sonnet generation task*. As with paraphrase detection, we implemented the DPO loss directly from the objective equation, which was fairly straightforward. Most of the effort in this task went into constructing the pairwise data, experimenting with different representations in order to improve performance metrics. We trained a SFT sonnet model to initialize the policy model and to serve as the frozen reference model.

The most effective pairwise data construction we tried uses automatically generated negatives stored in a JSONL file containing the fields `prompt`, `chosen`, and `rejected`, later using these pairs for DPO training. To construct this dataset, we first split each training sonnet into a prompt consisting of the first three lines and a ground-truth continuation consisting of the remaining lines – this continuation went in the "chosen" field. We then used the SFT sonnet model to generate a continuation from the same prompt, which went in the "rejected" field. An example of a generated pair can be found in Appendix A.1. This acts as an automated heuristic for quality ranking: SFT-generated continuations are in fluent English but often contain common autoregressive failures such as repetition or incoherent flow, making them lower quality than the preferred continuation. Our earlier attempts included simple

heuristic modifications, such as swapping two lines or dropping or replacing lines. While these methods produced plausible continuations, they were too similar to the true continuation, making the preference signal weak and resulting in less performance gains.

4.4 Smoothness-Inducing Adversarial Regularization

To improve the robustness and generalization of our models, we implement Smoothness-Inducing Adversarial Regularization (SIAR). The core objective of SIAR is to ensure that for some model input x , the model’s output distribution stays within a small neighborhood around x . We define the total training loss as:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{task}}(\theta) + \lambda_s \mathcal{R}_s(\theta)$$

Where $\mathcal{L}_{\text{task}}(\theta)$ is the standard cross-entropy loss, λ_s is a scalar regularization weight, and $\mathcal{R}_s(\theta)$ is the smoothness term. For some model $f(x; \theta)$, that smoothness term is defined using the Kullback-Leibler (KL) divergence between original model output and model output from adversarially-perturbed input embeddings. For some adversarial perturbation δ :

$$\mathcal{R}_s(\theta) = \max_{\|\delta\| \leq \epsilon} \text{KL}(f(x; \theta) \parallel f(x + \delta; \theta))$$

In our implementation, we apply perturbations to word embedding layer. We treat finding the optimal δ as an inner optimization problem, which we approximate using the single-step gradient method below:

$$\delta = \epsilon \cdot \text{sign}(\nabla_{\delta} \text{KL}(f(x; \theta) \parallel f(x + \delta; \theta)))$$

We ensure that the training is "leak-proof" (preventing the optimization above from polluting model weight gradients) by computing the gradient using `torch.autograd.grad`. For the paraphrase-detection task, the KL divergence is computed over the 2D logit distribution of the "yes/no" tokens. For the sonnet generation task, which generates longer token sequences, we use SIAR loss normalized by output length. Given the attention mask M and sequence length T , the sequence-level regularization is averaged across the non-padding tokens:

$$\mathcal{R}_{\text{seq}}(\theta) = \frac{1}{\sum_{t=1}^T M_t} \cdot \left(\sum_{t=1}^T M_t \cdot \text{KL}(f(x; \theta)_t \parallel f(x + \delta; \theta)_t) \right)$$

This normalization prevents the adversarial noise from dominating the modeling objective function in longer poems. This maintains consistent regularization strength across varying sequence lengths.

5 Experiments

5.1 Data

For the *sentiment analysis task*, which involves classifying the polarity of a text, we utilize the Stanford Sentiment Treebank (SST) and the CFIMDB datasets. The SST dataset consists of 11,855 movie review sentences annotated with five fine-grained sentiment labels: negative, somewhat negative, neutral, somewhat positive, or positive [17]. The CFIMDB dataset includes 2,434 movie reviews labeled as positive or negative. For the *paraphrase detection task*, which evaluates whether different phrases have the same semantic meaning, we use a subset of the Quora dataset [18]. The Quora dataset consists of 400,000 question pairs with labels indicating whether particular instances are paraphrases of one another. We applied data augmentation to the training set, doubling the size of the training set by switching the sentences (assuming paraphrases are a symmetrical relationship). For the *sonnet generation task*, we use a small dataset consisting of 154 sonnets written by William Shakespeare [16]. Each sonnet consists of 14 lines of text following the rhyme scheme ABAB CDCD EFEF GG, where lines that are assigned the same letter ID rhyme with each other.

5.2 Evaluation method

For the *sentiment analysis task*, the evaluation method was accuracy, which is a proportion of correctly predicted sentiment labels across the five classes for the SST dataset and the two binary classes for

the CFIMDB dataset. The *paraphrase detection task* also uses accuracy as the main evaluation metric, which is calculated by finding the proportion of correctly predicted "yes" or "no" tokens indicating whether one sentence is a paraphrase of another. For the *sonnet generation task*, the generated text is evaluated using the chrF metric, which measures how closely the generated language resembles original sonnets.

5.3 Experimental details

5.3.1 Baseline Implementation

For the *sentiment analysis task*, we evaluated two distinct training configurations across the SST and CFIMDB datasets. The last-linear-layer fine-tuning was conducted for 10 epochs with a batch size of 32 and a learning rate of $1e-3$, taking approximately 20 minutes per dataset on a 14-core Apple M3 GPU. The full-model fine-tuning was run for 10 epochs with a reduced batch size of 8 and a lower learning rate of $1e-5$, requiring 20–25 minutes per dataset on an NVIDIA T4 GPU. For the *paraphrase detection task*, the baseline GPT-2 model was fine-tuned on the Quora dataset for 3 epochs with a batch size of 8, a learning rate of $1e-5$, and 4 gradient accumulation steps. The total training and validation time was under 1.5 hours using an NVIDIA RTX 3500 Ada Generation GPU (12GB VRAM). Finally, for the *sonnet generation task*, the baseline GPT-2 model was trained in full fine-tune mode for 3 epochs with a batch size of 8. This training utilized a learning rate of $1e-5$ and took approximately 10 minutes to complete on an NVIDIA T4 GPU.

5.3.2 Parameter-Efficient Fine-Tuning with LoRA

For the *paraphrase detection task*, we evaluated two LoRA fine-tuning configurations on the Quora dataset using an NVIDIA L4 GPU. First, we applied LoRA ($r = 32$, $\alpha = 64$, scaling factor $\alpha/r = 2$) to the query and value modules across 48 layers of a GPT-2 Medium model (3.15M trainable parameters, 0.88% of total). We trained for 10 epochs using the AdamW optimizer with a batch size of 16 and a learning rate of 1×10^{-4} , taking approximately 10 hours total. Second, we applied LoRA with identical hyperparameters ($r = 32$, $\alpha = 64$) across 216 layers of a GPT-2 Large model, targeting the query, key, value, attention_dense, interim_dense, and out_dense modules (26.5M trainable parameters, 3.31% of total). This larger model was trained for 5 epochs with a reduced batch size of 4 and a learning rate of 5×10^{-5} , requiring roughly 20 hours total.

For the *sonnet generation task*, we also applied two LoRA configurations using an NVIDIA T4 GPU. We first fine-tuned a GPT-2 Medium model, applying LoRA ($r = 4$, $\alpha = 8.0$, scaling factor $\alpha/r = 2.0$) resulting in 1.77M trainable parameters (0.49% of total). The model was trained for 15 epochs with a batch size of 8 and a learning rate of 5×10^{-5} using the AdamW optimizer, completing in 10 minutes. Finally, we applied identical LoRA hyperparameters to a GPT-2 Large model, which yielded 3.32M trainable parameters (0.43% of total). This model was trained for 10 epochs with the same batch size of 8 and learning rate of 5×10^{-5} , also completing in approximately 10 minutes.

5.3.3 Direct Preference Optimization (DPO)

For the *paraphrase detection task* on the Quora dataset, we first ran the DPO implementation on the GPT-2 Base model, with full fine tuning on 124M parameters. The model was trained for 10 epochs with DPO $\beta = 0.1$, learning rate of 1×10^{-5} , batch size of 8, and the AdamW optimizer. On an NVIDIA L4 GPU, this task took 11.5 hours total. We also applied the same procedure to a GPT-2 Medium model with 355M parameters, training for 5 epochs with the same hyperparameters. This experiment ran on the same L4 GPU and completed in approximately 12.25 hours.

For the *sonnet generation task*, we applied DPO fine-tuning to GPT-2 Base and Medium initialized from a SFT sonnet model. For both GPT-2 Base and Medium, training was performed with DPO coefficient $\beta = 0.05$, learning rate 5×10^{-7} , batch size of 2, and the AdamW optimizer on an NVIDIA L4 GPU. We trained for 5 epochs, which took 10 minutes.

5.3.4 Smoothness-Inducing Adversarial Regularization (SIAR)

For the *paraphrase detection task*, we applied SIAR to the GPT-2 Base model. We used a perturbation budget of $\epsilon = 0.01$ and regularization weight of $\lambda_s = 0.5$. The model was trained for 5 epochs using symmetric data augmentation, batch size of 16, a gradient accumulation step of 2, and learning rate

of $1 \cdot 10^{-5}$. The training was performed on an NVIDIA RTX 3500 Ada GPU, taking 1.5 hours per epoch, around 7.5 hours in total.

For the *sonnet generation task*, we increased the perturbation budget to $\epsilon = 0.05$, and kept the regularization weight $\lambda_s = 0.5$. The model was trained for 20 epochs (although dev CHRF was maximized at 9 epochs) with batch size 8 and a reduced learning rate of $5 \cdot 10^{-5}$ to ensure stable convergence of the length-normalized KL divergence term. The training was performed on an NVIDIA RTX 3500 Ada GPU, taking about 20 seconds per epoch, about 7 minutes in total.

5.4 Results

Table 1 presents the quantitative results of our experiments across the sentiment analysis, paraphrase detection, and sonnet generation tasks. We compare our baseline implementations against the provided staff baselines and detail the performance improvements achieved through our extensions: LoRA, DPO, and SMART Regularization.

Approach	Sentiment (Acc)		Paraphrase (Acc)		Sonnet (chrF)	
	SST (Dev)	CFIMDB (Dev)	Dev	Test	Dev	Test
Staff Baseline (GPT2-Base, Last Linear Layer)	0.462	0.861	–	–	–	–
Our Baseline (GPT2-Base, Last Linear Layer)	0.467	0.865	–	–	–	–
Staff Baseline (GPT2-Base, Full Model)	0.513	0.976	0.882	0.856	41.414	41.208
Our Baseline (GPT2-Base, Full Model)	0.513	0.976	0.886	–	41.893	–
PEFT with LoRA (GPT2-Medium)	–	–	0.899	–	42.436	39.728
PEFT with LoRA (GPT2-Large)	–	–	0.908	0.894	41.948	–
Direct Preference Optimization (GPT2-Base)	–	–	0.888	–	41.445	–
Direct Preference Optimization (GPT2-Medium)	–	–	0.900	0.899	43.042	42.636
SMART Regularization (GPT2-Base)	–	–	0.897	0.895	42.279	41.849

Table 1: Quantitative results comparing our model’s performance against the established baselines across three tasks: sentiment analysis, paraphrase detection, sonnet generation.

5.4.1 Baseline Implementation

Our baseline implementation showed strong foundational performance across all tasks, validating our approach and aligning with our expectations. For *sentiment analysis*, transitioning from last-layer fine-tuning to full-model adaptation results in significant improvements, jumping from 0.467 to 0.513 accuracy on SST and 0.865 to 0.976 on CFIMDB. These results demonstrate that updating the entire network’s parameters, rather than just the final classification head, allows the model to learn the task-specific nuances. Furthermore, full-model fine-tuning showed a strong performance on the *paraphrase detection* task, achieving an accuracy of 0.886 and outperforming the staff baseline of 0.882. Similarly, our baseline exceeded the performance of the staff on the *sonnet generation* task, attaining a chrF score of 41.893 compared to 41.414. Overall, these results suggest that our baseline model adapted well to the downstream tasks, providing a solid benchmark for our following extensions.

5.4.2 Parameter-Efficient Fine-Tuning with LoRA

The implementation of Low-Rank Adaptation (LoRA) enabled us to leverage larger and more capable base models, such as GPT-2 Medium and Large, while maintaining computational efficiency. As detailed in the Appendix A.2, the fine-tuning process for these models was highly stable with smoothly decreasing training loss across all runs. On the paraphrase detection task, LoRA-adapted GPT-2 Medium achieved a development accuracy of 0.899, while scaling to GPT-2 Large pushed development accuracy to 0.908. This shows that larger models capture more complex semantic representations during pre-training, which significantly benefits downstream tasks even when adapting a minimal fraction of parameters.

However, the sonnet generation task highlighted the challenges of applying these large methods to extremely constrained domains with limited data. While the LoRA-adapted GPT-2 Medium achieved a strong development chrF score of 42.436, the test score dropped to 39.728, which is lower than the full-model baseline’s test score. Scaling to GPT-2 Large with LoRA yielded a development chrF of 41.948. Despite applying LoRA with a low rank intended to prevent the model from memorizing the training data and overfitting, the performance gap suggests the models still over-optimized on the limited set of 143 training sonnets, struggling to generalize their stylistic adaptations to unseen text.

5.4.3 Direct Preference Optimization (DPO)

Given that DPO is designed to learn preferences between alternative outputs, it is naturally better suited for generative tasks than classification tasks. Thus, we hypothesized that it would make significant improvement over the baseline in sonnet generation, and slight to moderate improvement in paraphrase detection. Since DPO requires full fine-tuning of all model parameters, scaling to GPT-2 Large exceeded our GPU memory constraints, so we limited our experiments to GPT-2 Base and Medium.

On the paraphrase detection task, DPO applied to GPT-2 Medium results in development accuracy of 0.900 and test accuracy of 0.899. The DPO training dynamics exhibited stable convergence, characterized by a monotonically decreasing loss and a consistent improvement in development accuracy across all epochs (A.3). As expected, the larger model achieved higher accuracy due to its greater representational capacity, allowing it to capture better semantic relationships between sentence pairs. Even though DPO is not suited for tasks containing high-quality binary labels, it still produced a small improvement over the baseline, likely because the preference pairs enable the model to more strongly separate correct and incorrect outputs.

For the sonnet generation task, the capabilities of DPO are more evident. On GPT-2 Medium, we reached a development chrF score of 43.042 and test chrF score of 42.636. We compared outputted generated sonnets for the baseline and DPO, noticing that baseline outputs frequently contained corrupted tokens and the poems did not make sense. In contrast, the DPO model outputs have more consistent line formatting and are more on-theme with the prompt provided in the first three lines.

5.4.4 Smoothness-Inducing Adversarial Regularization (SIAR)

The implementation of SIAR provided measurable improvements in both task performance and model robustness. On the *paraphrase detection task*, SIAR achieved a dev accuracy of 0.897, a significant improvement over our baseline of 0.886. This gain suggests that enforcing local smoothness helps the model focus on semantic meaning rather than syntactic noise in the prompt.

We also evaluated the "brittleness" of our models by measuring the adversarial flip rate: the frequency at which a minimal, meaning-preserving nudge ($\epsilon = 0.01$) in the embedding space causes the model to change its prediction. Our baseline model exhibited high brittleness with 48% flip rate. Following 3 epochs of SIAR training (the same number of epochs as our baseline received) this rate dropped to 25%, and further dropped to 24% after 5 epochs of training. The two-fold reduction in brittleness confirms that SIAR successfully smooths the jagged decision boundaries, producing a more robust classifier.

In the *sonnet generation task*, SIAR reached a dev set chrF score of 42.279, also improving over our baseline of 41.893. The length-normalization of SIAR loss was critical for this task, as otherwise, the total adversarial penalty on longer (e.g. 128 token) sequences dominated the language modeling signal. These results indicate that preference optimization was superior for steering model behavior, while SIAR was more effective for stabilizing the underlying manifold.

6 Analysis

Our results indicate that while all three extensions (LoRA, DPO, and SIAR) improved upon the full-model fine-tuning baseline, they yielded different advantages depending on the task and model scale. In the *paraphrase detection task*, LoRA allowed us to adapt GPT-2 Large within our compute constraints, yielding the highest dev accuracy of 0.908 and exhibiting stable training curves A.2. This demonstrates that increased representational capacity is a primary driver of semantic classification performance. Specialized objectives demonstrated that smaller architectures with complex learning

objectives can serve as robust alternatives to model scaling. For an example, see the comparison of model outputs for a paraphrase detection task in the Appendix A.6. DPO on GPT-2 Medium achieved 0.900, and SIAR on GPT2-Base achieved 0.897, both outperforming our baseline of 0.886. Both approaches also had minimal degradation on the test sets.

In the *sonnet generation task*, we observed a distinction between steering model behavior and stabilizing the underlying manifold. DPO achieved the highest dev chrF score of 43.042, by explicitly training the model to prefer real Shakespearean continuations over baseline-generated alternatives. This approach seems most effective for capturing the stylistic constraints of sonnets, as its objective is uniquely suited for guiding the model towards a target style. For an example, see the comparison of all approaches on sonnet #11 on the dev set in Appendix A.5. In contrast, SIAR focused on local stability, reaching a dev chrF score of 42.279. While slightly lower than DPO’s peak performance, SIAR’s performance demonstrates that enforcing smoothness in the embedding space is a viable approach for improving output quality in low-data regimes. The noise in the dev chrF observed across SIAR training highlights that there is inherent volatility in sampling-based metrics in text generation (even though the training loss was decreasing smoothly).

Finally, our analysis suggests that different extension methods optimize different aspects of model behavior. While LoRA and DPO focused on maximizing task-specific scores, SIAR provided a unique secondary benefit of improving reliability of the model’s decision boundaries. Our evaluation of the adversarial flip rate showed that the baseline model was susceptible to prediction flips 48% of the time under microscopic embeddings noise. The application of SIAR successfully reduced this brittleness by 2x. While LoRA remains the superior choice for maximizing accuracy through scale, specialized objectives like DPO and SIAR offer critical tools for behavior alignment and structural stability.

7 Conclusion

In this project, we successfully implemented a baseline GPT-2 model and explored three advanced fine-tuning extensions – Low-Rank Adaptation (LoRA), Direct Preference Optimization (DPO), and Smoothness-Inducing Adversarial Regularization (SIAR) – across the downstream tasks of sentiment analysis, paraphrase detection, and sonnet generation. Our main finding is that each method offers different advantages depending on the target task and model scale. LoRA proved highly effective for semantic classification, allowing us to leverage a GPT-2 Large architecture within compute limits to achieve our highest paraphrase detection development accuracy of 0.908. At the same time, DPO excelled in generative tasks, reaching a peak development chrF score of 43.042 by effectively steering the model to adhere to Shakespearean stylistic constraints. Furthermore, SIAR reduced our baseline model’s adversarial flip rate from 48% to 24%, highlighting its value in improving the reliability of decision boundaries.

The main limitation of our study was the significant computational bottleneck associated with full-model adaptation techniques like DPO, which restricted our ability to scale these specific optimizations to the GPT-2 Large architecture. Similarly, SIAR was only evaluated on GPT-2 Base due to its computational overhead from the adversarial perturbation and KL divergence computation at each training step, preventing us from determining whether its robustness benefits scale with model size or whether larger models are inherently less brittle. Future work could explore scaling DPO to GPT-2 Large and evaluating SIAR on GPT-2 Medium and Large to better understand how these methods behave at greater model capacity. Additionally, future work could explore combining these extensions, such as applying DPO on top of a LoRA-adapted model or integrating SIAR during preference optimization, to simultaneously leverage parameter efficiency, target style alignment, and boundary robustness.

Team contributions

All team members contributed equally to implementing the GPT-2 model and the sentiment analysis task. Each team member was responsible for implementing, testing, and writing about one extension: Anastasiya for PEFT with LoRA, Eva for DPO, Michael for SMART regularization. All team members contributed equally to writing the milestone report, final report, and poster.

References

- [1] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.
- [2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [3] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*, 2023.
- [4] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.
- [5] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, 2021.
- [6] 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. In *International Conference on Learning Representations*, 2022.
- [7] Rahul Bhagat and Eduard Hovy. What is a paraphrase? *Computational Linguistics*, 39(3):463–472, 2013.
- [8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019.
- [9] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics.
- [10] Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. Deep-speare: A joint neural model of poetic language, meter and rhyme. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1948–1958, Melbourne, Australia, July 2018. Association for Computational Linguistics.
- [11] 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, 2024.
- [12] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.
- [13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- [14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

- [15] 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 softness-induced adversarial regularization and bound optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2189, 2020.
- [16] CS224N Course Staff. Cs 224N: Default final project: Build GPT-2. Stanford University CS224N Course Handout, 2026.
- [17] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.
- [18] Quora. First quora dataset release: Question pairs. <https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs>, 2017.

A Appendix

A.1 Appendix: Example Sonnet DPO Training Pair

Below is one example pair used for DPO training in the sonnet generation task. The prompt consists of the first three lines of a Shakespeare sonnet. "Chosen" is the true Shakespearean sonnet continuation, and the "rejected" continuation is generated by the baseline SFT sonnet model. The generated continuation is in fluent English and uses poetic language, but in this example it does not flow with the sonnet's theme and the writing style is less creative (almost every line starting with "And").

```

1 {
2   "prompt": "Then hate me when thou wilt, if ever, now,\nNow, while
3   the world is bent my deeds to cross,\nJoin with the spite of
4   fortune, make me bow,\n",
5   "chosen": "And do not drop in for an afterloss.\nAh, do not, when my
6   heart hath 'scaped this sorrow,\nCome in the rearward of a conquered
   woe;\nGive not a windy night a rainy morrow,\nTo linger out a
   purposed overthrow.\nIf thou wilt leave me, do not leave me
   last,\nWhen other petty griefs have done their spite,\nBut in the
   onset come; so shall I taste\nAt first the very worst of fortune's
   might;\nAnd other strains of woe, which now seem woe,\nCompared with
   loss of thee will not seem so.",
   "rejected": "In so quiet a time, when my chaste body is done,\nAnd
   now I rise, and swear to the ancient age,\nAnd swear that my love
   for thee is sweet,\nAnd that I may be in thee in thy art,\nAnd
   how much of thy love is thou,\n    I have no desire to change,",
   "sonnet_id": 89
}
```

A.2 Appendix: Training Dynamics of LoRA Fine-Tuning

A.2.1 Paraphrase Detection Training Curves

Figure 1 shows the learning trajectories for both medium and large GPT-2 architectures. Both models exhibit a smooth, monotonically decreasing cross-entropy training loss. Additionally, the development accuracy steadily climbs and plateaus without significant volatility, reaching peak accuracies of 0.899 and 0.908, respectively. This confirms stable convergence during the adaptation of LoRA on GPT-2 Medium and GPT-2 Large architectures.

A.2.2 Sonnet Generation Training Curves

Figure 2 shows the training loss trajectories for the sonnet generation task. Despite the small size of the training dataset, the cross-entropy loss curves for both the GPT-2 Medium and Large models

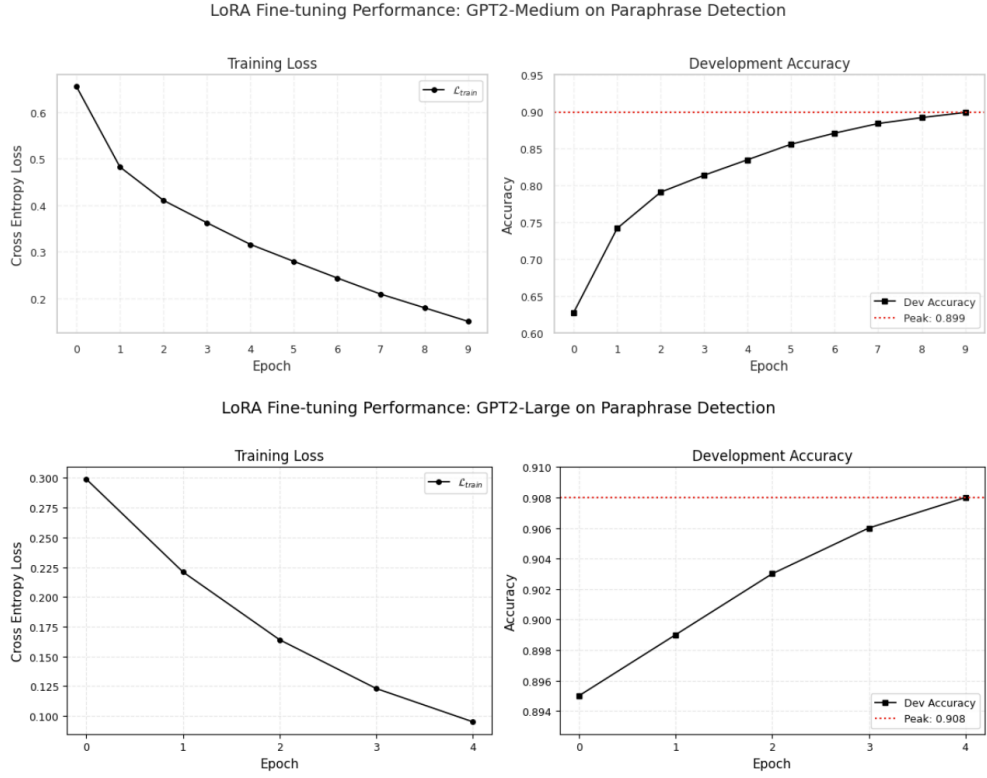


Figure 1: Training loss and development accuracy per epoch for GPT-2 Medium and GPT-2 Large on the *paraphrase detection* task.

show consistent, stable convergence over their respective 15 and 10 epoch training runs. These learning curves confirm that our selected LoRA hyperparameters prevented overfitting within the highly constrained sonnet generation domain.

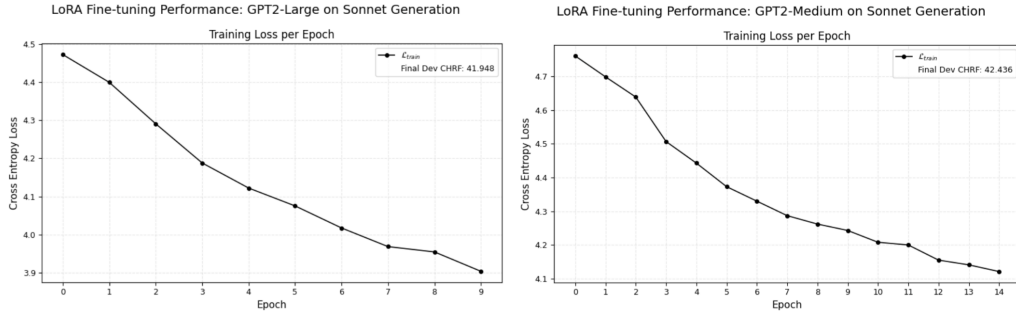


Figure 2: Training loss per epoch for GPT-2 Large and GPT-2 Medium on the sonnet generation task.

A.3 Appendix: Training Dynamics of DPO

A.3.1 Paraphrase Detection Training Curves

Figure 3 shows the training loss and accuracy curves for DPO on the GPT-2 Medium model. Across the 5 epochs, there was smooth and consistent training.

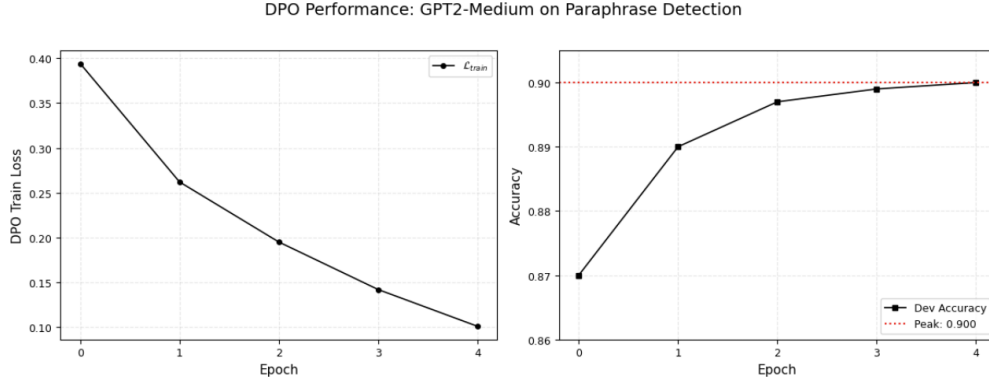


Figure 3: Training loss and dev accuracy per epoch for DPO on GPT-2 Medium (paraphrase).

A.3.2 Sonnet Generation Training Curves

We captured the loss curve during the Sonnet Generation task using DPO on GPT-2 Medium, depicted in Figure 4. Across 5 epochs, there was a stable decrease in train loss.

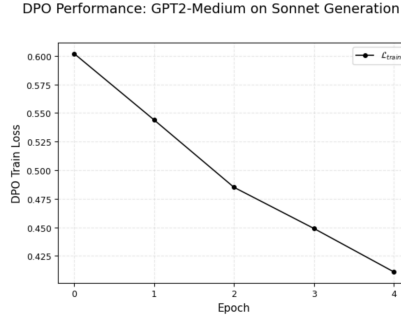


Figure 4: Training loss per epoch for DPO on GPT-2 Medium (sonnet).

A.4 Appendix: Training Dynamics of SIAR

A.4.1 Paraphrase Detection Training Curves

Figure 5 depicts the train and dev accuracy on the paraphrase detection task, training over 5 epochs with Smoothness-Inducing Adversarial Regularization. The accuracy steadily increases over the 5 epochs, and the dev accuracy seems to plateau. The loss also steadily decreases over the 5 epochs (Figure 6).

A.4.2 Sonnet Generation Training Curves

Figure 7 depicts the training loss and Dev CHRF during the training job for sonnet generation, over 9 epochs (using Smoothness-Inducing Adversarial Regularization). The loss steadily decreases throughout the training run. The Dev CHRF hovers around 42, but generally increases throughout the training.

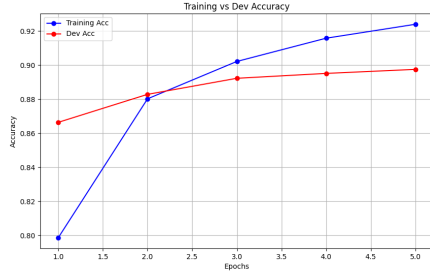


Figure 5: Train and Dev accuracy on paraphrase detection with SIAR over 5 epochs.

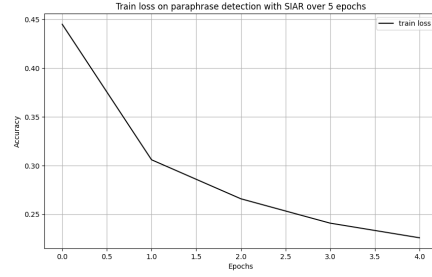


Figure 6: Training loss on paraphrase detection with SIAR over 5 epochs.

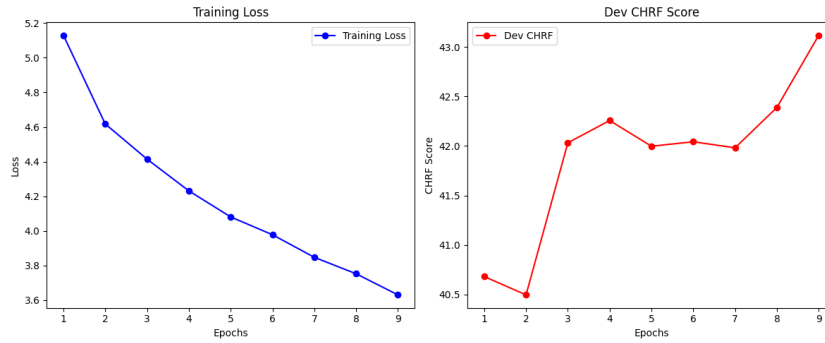


Figure 7: Training loss and Dev CHRF for the Sonnet Generation task using SIAR across 9 epochs.

A.5 Appendix: Sonnet generation comparison across all approaches

Below A.5 are the best models from each extension, all generating a sonnet from the same starting prompt (dev set prompt #11). LoRA produced words that do not exist (ailbut, unfostuate). DPO uses poetic, Shakespeare-like language but is cut off at the

end and does not follow the prompt’s narrative. SIAR also deviates from the narrative.

LoRA best:	DPO best:	SIAR best:
Lo, as a careful huswife runs to catch One of her feathered creatures broke away, Sets down her babe, and makes all swift dispatch And hoards accordingly if their weather reign With dangers to unfostuate homes overrange, To despatch the offending cut-tred figure for more; And scarce let being dwell, without herd slew Or ailbut (ex-red blood have its part) frequent dreams, For men descending from the field straight avoid Autumn once pursuing half the foolish sufficiency that They fail towards and tho for a loss worse to bear, For knowledge hath taught those who so wit find The wastiness they call breath that restoreth which given For fear burns from whose howt sleep will rest As nights	Lo, as a careful huswife runs to catch One of her feathered creatures broke away, Sets down her babe, and makes all swift dispatch To hide her injuries from the most earnest hunters, O’erskill her lambs and kill her foals without fear; Profitful wretches of blood, whose faces in the dust so lie, Unless they are camest away to do violence’s rhyme, To ransom others with such graces o’er, Such as their master should’st so render. So a slight neglect or unscrupulous son, As it turns, as pale a prey, says sin, Making for his captive this most imprecation: Pitying loss thou art to them dull rear,	Lo, as a careful huswife runs to catch One of her feathered creatures broke away, Sets down her babe, and makes all swift dispatch. She leads him twice to her door and leads him away, Just as though she were a condemned murderess, She gives him another moment’s warning, And forwards onwards his dressing foot Bavity horses with boot." She fled when thou wert come At the moving of her pursuing wolf And swift horses at her journey wearied, And so she did bear him with speed, Upon her ride moving again her offenders from right. This she did devise for the child, To free him of her woe or give him travel. For having three compare with one

A.6 Appendix: Paraphrase Detection comparison across all approaches

Sentence 1: What are the best ways to lose weight?

Sentence 2: What are some tips for losing weight?

Full Prompt: Is “{S1}” a paraphrase of “{S2}”? Answer “yes” or “no”:

Model Approach	Prediction	Token ID	Status
Ground Truth	YES	–	–
PEFT (LoRA-Large)	NO	3919	Incorrect
DPO (Medium)	YES	8505	Correct
SIAR (Base)	YES	8505	Correct

Table 2: Model predictions on a pair with high semantic similarity but significant lexical variation.

The phrases "best ways" and "tips" are semantically similar in this context, but they are different lexically, because both different nouns and adjectives are used in the two sentences. Whether the model makes a mistake in this context depends more on the "brittleness" of the model, which DPO and SIAR effectively address. The failure of the PEFT + LoRA model in this case suggests that scaling parameter count alone is insufficient to overcome brittleness in vocabulary.