

Preference-Optimized GPT-2 for Cloze-Style Paraphrase Detection and Sonnet Generation via DPO with GPT-Scored Pairs

Stanford CS224N Default Project

Arun Moorthy
Department of Computer Science
Stanford University
amoorth2@stanford.edu

Welton Wang
Department of Computer Science
Stanford University
welton@stanford.edu

Andrew Park
Department of Computer Science
Stanford University
aspark@stanford.edu

Abstract

Fine tuning large language models for downstream tasks is super common, but the choice of optimizer and training objective can really affect performance. In this project, we fine tuned GPT-2 for two tasks: paraphrase detection on the Quora Question Pairs dataset and Shakespearean sonnet generation. We investigate two areas of improvement over a baseline supervised fine tuning (SFT) pipeline with AdamW optimizer. First, we compare three optimizers under SFT, which are AdamW, SOAP (a preconditioned variant of Adam), and Muon (an optimizer based on Newton Schulz orthogonalization). Second, we tried Direct Preference Optimization (DPO) on top of SFT checkpoints to align model outputs with human preferences. Our results show that DPO improves sonnet generation chrF from 39.284 to 41.858, but surprisingly underperforms SFT on paraphrase detection (0.875 vs 0.895). Among the SFT optimizers, Muon slightly outperforms AdamW (0.897 vs 0.895 paraphrase dev accuracy), while SOAP slightly underperforms AdamW despite a lot of hyperparameter tuning. These findings suggest that DPO's effectiveness is task dependent, and that optimizer choice produces smaller effects than expected in the fine tuning setting.

1 Key Information

- TA mentor: Simon Kim
- External collaborators: No
- External mentor: No
- Sharing project: No

2 Introduction

Pre-trained transformer language models such as GPT-2 [1] have become foundational building blocks for natural language processing tasks. The standard approach for adapting these models involves supervised fine tuning (SFT), where the pre-trained weights are updated on labeled data. While this approach works, there are several design choices that can affect the final performance including the optimizer, the learning rate, and the training objective itself.

In this project, we looked at these choices in the context of two tasks from the CS224N default final project, which are paraphrase detection, a classification task on the Quora Question Pairs dataset, and sonnet generation, a language modeling task on Shakespeare’s sonnets. Our base setup fine tunes a GPT-2 model using AdamW [2], which is our baseline.

We explored two main directions for improvement. The first direction is optimizer ablation, meaning that we compared AdamW against two recently proposed optimizers, SOAP [3] and Muon [4]. The second main direction is Direct Preference Optimization (DPO) [5], which is a training objective that directly optimizes a policy to prefer high quality outputs over low quality ones, without requiring a different reward model. DPO uses paired preference data (preferred vs dispreferred completions) and adjusts the model’s log probabilities to increase the margin between preferred and dispreferred outputs. This is all relative to a frozen reference model.

Our key findings from our experiments are that DPO improves sonnet generation quality but actually is worse for paraphrase detection accuracy compared to SFT alone. This was decently unexpected. In the optimizer comparison, Muon slightly outperformed AdamW in both tasks, while SOAP slightly underperforms AdamW even though we tried tuning our hyperparameters a lot.

3 Related Work

Language model fine tuning. GPT-2 showed us that large autoregressive language models can learn useful information to help with downstream tasks. Fine tuning pretrained transformers has become the dominant idea for NLP with approaches ranging from full model fine tuning to parameter efficient methods like LoRA [6]. In this project, we fine tuned all parameters of GPT-2.

Optimizer design. Adam [7] and its weight decay variant AdamW are the main optimizers for transformer training. However, there are a few papers that suggest other ones. Shampoo [8] uses Kronecker factored preconditioning of the gradient covariance matrix. SOAP brings this preconditioning into Adam’s eigenbasis for improved convergence. Muon takes a slightly different approach, which has shown promising results for pre-training. However, we were not sure on the effectiveness of these optimizers for fine tuning (as opposed to pre-training).

Preference optimization. Reinforcement Learning from Human Feedback (RLHF) aligns language models with human preferences by training a reward model and then optimizing the policy with PPO [9]. DPO simplifies this by directly optimizing the policy on preference pairs without a separate reward model. The DPO objective increases the log probability of preferred completions relative to dispreferred ones, which is subject to a KL penalty against a reference model. DPO has been applied successfully to many different tasks and showed a lot of promise in literature. We wanted to extend it to our tasks, which are paraphrase detection and sonnet generation.

Paraphrase detection and sonnet generation. Paraphrase detection seems to be a well studied task. The Quora Question Pairs dataset provides labeled question pairs showing semantic equivalence. Sonnet generation is a constrained creative writing task requiring the model to generate 14 line poems that follow Shakespearean conventions. Both tasks test different capabilities of the fine tuned model, including classification accuracy and long form generation quality.

4 Approach

4.1 Base Model and SFT

We use GPT-2 initialized from pre-trained weights from HuggingFace. The model architecture has 12 transformer layers with 768 dimensional hidden states, 12 attention heads, and a vocabulary of 50,257 BPE tokens. We used the starter code to implement the base GPT implementation. We tested it using the sanity checks that were provided to us.

Sentiment classification baseline. Before moving to extension tasks, we verified our GPT-2 implementation by training a sentiment classification baseline. We add a classification head over the last token’s hidden state representation. In the last linear layer setting, all GPT-2 parameters are frozen and only the classification head is trained with cross entropy loss. This serves as a sanity check that our GPT-2 produces meaningful representations.

Paraphrase detection. We frame paraphrase detection as a cloze style next token prediction task. Given two questions q_1 and q_2 , we construct the input: 'Question 1: "{q1}" Question 2: "{q2}" Are these questions asking the same thing?'. Rather than using a separate classification head, we use weight tied token prediction via logits $= h_{\text{last}} \cdot E^\top$, where E is the word embedding matrix. The model predicts the next token after this prompt. We look at the logits at the token indices for “yes” and “no”, and take the argmax over the full vocabulary which is our prediction. The training loss is cross entropy over the full vocabulary at the last token spot.

Sonnet generation. For sonnet generation, we fine tune GPT-2 using standard autoregressive language modeling. The model receives a full sonnet as input and is trained to predict each next token using cross entropy loss over the entire sequence. At inference time, we provide the first three lines of a held out sonnet as a prompt and generate the remaining lines using top p sampling. We use temperature = 1.2 and top_p = 0.9.

4.2 Optimizers

One main question in this paper is whether the choice of optimizer during SFT can actually change downstream task performance. We compare three optimizers that span different points in the design space, including AdamW as the well established default, SOAP as a somewhat similar one, and Muon as a fundamentally different optimizer. All three optimizers are applied under identical conditions so that any performance differences can be attributed to the optimizer itself.

AdamW. AdamW maintains exponential moving averages of the gradient and the squared gradient (second moment v_t). The bias corrected update rule is:

$$\theta_t = \theta_{t-1} - \alpha_t \cdot \frac{m_t}{\sqrt{v_t} + \epsilon} - \alpha \cdot \lambda \cdot \theta_{t-1}$$

where α_t has the bias correction terms, and the final term applies decoupled weight decay with λ . We implemented AdamW from scratch, which is also the auxiliary optimizer for the non Muon parameters in our Muon experiments and as the base optimizer for DPO training. Our implementation passed the provided optimizer unit tests, so that was correct.

SOAP. SOAP extends Adam by projecting gradients into the eigenbasis of a Shampoo Kronecker factored preconditioner before applying the standard Adam update. Essentially, prior to applying a gradient update, it attempts to project the gradients into a different space that is better aligned with the loss surface, allowing correlated parameters to have more similar scaling (instead of per-parameter in Adam).

So, for each weight matrix, SOAP keeps per axis covariance matrices GG_i that approximate the gradient second moments along each dimension. SOAP also recomputes orthogonal bases Q_i and then gradients are projected into this basis through, the Adam update is applied in the rotated space, and the result is projected back. Our implementation uses a precondition frequency of 10 steps, a maximum precondition dimension of 2048.

Muon. Muon takes a different approach to optimization. Instead of scaling gradients by estimates of curvature, Muon replaces the update direction entirely with the nearest orthogonal matrix to the momentum buffer. The Newton Schulz iteration repeatedly applies a polynomial update to the momentum matrix, converging to the nearest orthogonal matrix in approximately 5 iterations. The intuition for muon mostly comes from treating gradient updates across parameters not as individual, independent updates but instead an entire matrix that represents some direction(s) and magnitude.

Importantly, not all parameters are suitable for orthogonalized updates. Embedding layers including word_embeddings and pos_embeddings and task specific heads are excluded from the Muon path because they are either lookup tables or low dimensional projections where the orthogonalization assumption does not hold. These parameters are updated with a standard AdamW step using a separate learning rate. This creates a sort of mixed optimizer with two parameter groups, each with its own learning rate. The Muon learning rate is usually set much higher than the Adam learning rate.

4.3 Direct Preference Optimization (DPO)

DPO trains a policy model π_θ to prefer certain outputs over others by optimizing on preference pairs. Given a prompt x , a preferred completion y_w and a dispreferred completion y_l , the DPO loss is:

$$L_{DPO} = -E \left[\log \sigma \left(\beta \left(\log \frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)} \right) \right) \right]$$

where π_{ref} is a frozen copy of the SFT model and β controls the strength of the KL penalty. The policy model is initialized from the SFT checkpoint. The loss encourages the policy to increase the log probability ratio of preferred over dispreferred completions, relative to the reference model. All DPO code is implemented from scratch, reusing our existing AdamW optimizer and other utilities.

Preference pair construction. We construct preference pairs differently for each task:

For paraphrase detection, we use heuristic pairs derived from the gold labels. Given a question pair with ground truth label (e.g., “yes” for paraphrases), we set the preferred completion to the correct label token and the dispreferred completion to the incorrect label token. The prompt is the cloze style input that we described before. The DPO loss for paraphrase operates on the single next token logits, comparing the log probabilities of the preferred and dispreferred label tokens. Note that this technically becomes the SFT loss with an additional contrastive penalty.

For sonnet generation, we use the first three lines of each training sonnet as the prompt. The preferred continuation is the ground truth remaining lines. To generate dispreferred continuations, we sample from the SFT model at varying temperatures and also create heuristic negatives by truncating sonnets to only 5 lines or changing the line order of the continuation. This produces approximately two preference pairs per training sonnet. The DPO loss for sonnets operates on the full sequence log probabilities, adding token level log probs across the continuation. The reference model is a frozen copy of the SFT checkpoint.

5 Experiments

5.1 Data

We use the provided datasets with no external data. For sentiment classification baselines: SST (train 8,544 / dev 1,101, 5 class) and CFIMDB (train 1,701 / dev 245, binary). For paraphrase detection: Quora Question Pairs (train 141,506 / dev 20,215 / test 40,431). For sonnet generation: Shakespeare sonnets (train 143 / held out test 12 / held out dev 12). For DPO, preference pairs are created from the respective training sets as we just described.

5.2 Evaluation Metrics

For sentiment classification and paraphrase detection we report accuracy. For sonnet generation we report chrF [10] (character level F score).

chrF computes character precision and recall between the generated text and the reference, then combines them via the F score.

5.3 Experimental Details

All experiments use the GPT-2 base model. We trained on a single GPU (1xH200).

Sentiment baselines. Sentiment classification uses frozen GPT-2 parameters with a trainable classification head, batch size 64, learning rate 1e-3, dropout 0.3, for 10 epochs.

SFT hyperparameters. For paraphrase detection with AdamW, we use a learning rate of 1e-5, batch size of 32, and train for 10 epochs with full model fine tuning. For sonnet generation with AdamW, we use a learning rate of 1e-5, batch size of 8, and train for 10 epochs. For SOAP, we use a learning rate of 3e-5, precondition frequency of 10, and max precondition dimension of 2048. For Muon, we set the Muon learning rate to 20x the base Adam learning rate (clamped between 2e-4 and 2e-3), with momentum 0.95 and Nesterov momentum enabled. Embedding and head parameters use Adam with learning rate equal to the base rate and betas (0.9, 0.95).

DPO hyperparameters. We initialize the policy from the best SFT checkpoint. We use AdamW with learning rate $1e-6$, batch size 16 (paraphrase) or 4 (sonnet), and train for 3 epochs. The DPO temperature parameter β is set to 0.1. We apply gradient clipping with max norm 1.0. We ablate over $\beta \in \{0.05, 0.1, 0.2, 0.5\}$ and learning rate as one of these: $\{5e-7, 1e-6, 5e-6\}$.

Hyperparameter tuning effort. We performed a lot of hyperparameter tuning for all methods. For AdamW and Muon, we swept over learning rates and training epochs. For SOAP, we put a lot of effort into tuning by exploring different configurations of the learning rate and preconditioner settings. Despite these efforts, SOAP didn’t surpass AdamW in our fine tuning setting. We report the best results obtained for each optimizer after this parameter tuning.

5.4 Results

As mentioned before, we first verified our GPT-2 implementation using the task of sentiment classification before we did our extensions. Here are some of the results from that task.

Task	Fine Tune Mode	Dev Accuracy
SST (5 class)	last linear layer	0.468
SST (5 class)	full model	0.503
CFIMDB (binary)	last linear layer	0.861
CFIMDB (binary)	full model	0.980

Table 1: Sentiment classification baseline results.

Table 1 shows that our GPT-2 implementation produces meaningful representations with SST accuracy of 0.468 on 5 class classification with frozen parameters, improving to 0.503 with full model fine tuning. CFIMDB accuracy reaches 0.861 with frozen parameters and 0.980 with full model fine tuning. These are consistent with expectations and verify our implementation.

Method	Para. Dev Acc	Sonnet Dev chrF	Sonnet Test chrF	Para. Test Acc
SFT + AdamW (baseline)	0.895	39.284	—	—
SFT + Muon	0.897	39.928	—	0.896
SFT + SOAP	0.882	38.892	—	0.871
DPO (from AdamW SFT)	0.875	41.858	41.520	—

Table 2: Main results across all methods. Paraphrase accuracy and sonnet chrF on dev and test sets.

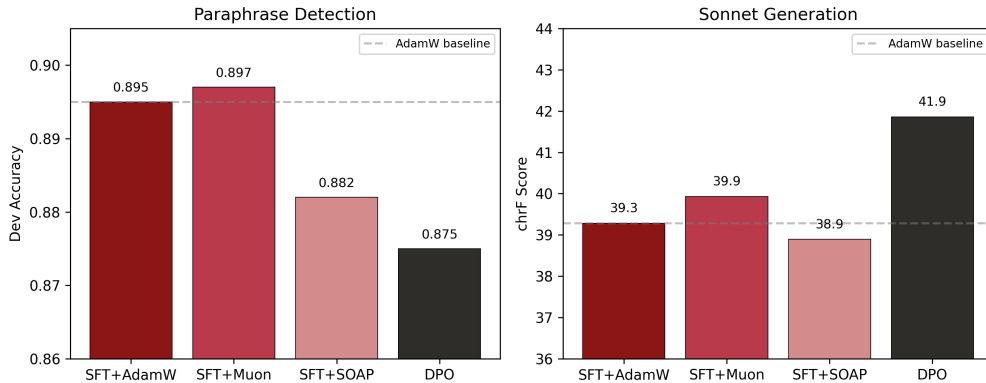


Figure 1: Performance comparison across all methods. Left: paraphrase detection dev accuracy; Right: sonnet generation chrF. DPO achieves the best sonnet chrF but the worst paraphrase accuracy, while Muon slightly outperforms the AdamW baseline on both tasks.

The results reveal an interesting split across tasks. On sonnet generation, DPO achieves the highest dev chrF of 41.858 and a test chrF of 41.520, improving over the SFT AdamW baseline of 39.284.

This confirms that preference based training helps the model produce higher quality continuations. But on paraphrase detection, DPO actually underperforms SFT, dropping from 0.895 to 0.875 on the dev set. This was pretty surprising to us.

Among SFT optimizers, Muon achieves the best results overall with 0.897 paraphrase dev accuracy and 39.928 sonnet chrF, both slightly above AdamW. SOAP underperforms AdamW on both tasks, with 0.882 paraphrase dev accuracy and 38.892 sonnet chrF.

6 Analysis

6.1 Optimizer Ablation Under SFT

To compare optimizers, we held everything constant except the optimizer and its hyperparameters, meaning we had the same GPT-2 base model, the same datasets, and the same evaluation pipeline.

Optimizer	Para. Dev Acc	Sonnet chrF	Δ vs. AdamW (Para.)
AdamW	0.895	39.284	—
Muon	0.897	39.928	+0.002
SOAP	0.882	38.892	−0.013

Table 3: Optimizer ablation under SFT.

Muon shows a slight improvement over AdamW, achieving 0.897 vs 0.895 on paraphrase dev accuracy and 39.928 vs 39.284 on sonnet chrF. More importantly though, we find that Muon converges on a higher dev accuracy at a significantly faster rate than AdamW, taking half the number of epochs (5 vs 10). The orthogonalized updates may help by maintaining well conditioned weight matrices during fine tuning, allowing the model to make more effective updates even with small learning rates. The improvement is small, suggesting that AdamW already performs reasonably well when fine tuning from a good initialization, but Muon extracts some additional value from its update transformation, and is significantly more compute-efficient.

SOAP underperforms AdamW in our fine tuning setting, scoring 0.882 vs 0.895 on paraphrase and 38.892 vs 39.284 on sonnet chrF, even though we spent time trying to optimize the hyperparameters a lot. We performed repeated sweeps over the SOAP learning rate, precondition frequency, and preconditioner dimension settings. While it is possible that a different configuration could result in stronger SOAP performance, in our best tuned runs AdamW remained more effective. So, we report SOAP as underperforming AdamW in this use case. Note that SOAP may be better in other use cases. SOAP’s preconditioning adds overhead in maintaining and recomputing the eigenbasis, and this additional complexity may not pay off in the fine tuning regime where the model is already near a good solution and trains for relatively few steps. The periodic eigenbasis recomputation may also introduce noise when the number of training steps is small relative to the precondition frequency.

6.2 DPO Hyperparameter Ablation

We ablated the DPO temperature β over $\{0.05, 0.1, 0.2, 0.5\}$ for sonnet generation and the learning rate over $\{5e-7, 1e-6, 5e-6\}$ for paraphrase detection. For learning rate, $1e-6$ seemed to be the best from what we tested.

6.3 Qualitative Observations on Sonnet Generation

On sonnet generation, where DPO does help (improving chrF from 39.284 to 41.858), we saw improvements in two common SFT failure modes: line repetition and incorrect line counts. The preference pairs penalize these issues because the not preferred (model generated) completions tend to be repetitive, but the preferred completions have the correct 14 line structure and no repetition. Rhyme adherence also tends to improve, as DPO steers the model toward ground truth completions that follow the correct scheme.

6.4 Observations on Paraphrase Detection

On paraphrase detection, we find that applying DPO on the SFT’ed model actually makes the model perform worse. It is uncertain as to why this is, but one potential hypothesis is that since paraphrase detection is a binary task, DPO over yes/no labels is essentially trying to maximize $\text{logp}(\text{correct}) - \text{logp}(\text{wrong})$, and SFT maximizes $\text{logp}(\text{correct})$. Therefore, this is somewhat similar to performing more SFT on the target labels, which could lead to overfitting and therefore worse dev performance.

7 Discussion

The most surprising result from our experiments is that DPO’s effectiveness is task dependent. On sonnet generation, DPO provides clear improvements over SFT (41.858 vs 39.284 chrF), which aligns with what we expected, showing DPO’s strength in generation tasks. On paraphrase detection, DPO actually hurts performance (0.875 vs 0.895). This suggests that the nature of the downstream task matters a great deal when deciding whether to apply preference optimization on top of SFT.

For generation tasks like sonnet completion, the preference pairs capture holistic qualities that the token level SFT loss cannot directly optimize for. The ground truth Shakespeare continuations are qualitatively different from model generated continuations in ways that DPO can learn from. For classification tasks like paraphrase detection, SFT already provides a direct signal for the correct answer, and DPO’s preference framing does not add useful information beyond what cross entropy already provides. In fact, the additional DPO training may degrade the model’s calibration by over specializing on the preference margin.

The optimizer comparison shows a smaller but more consistent pattern. Muon’s orthogonalized updates help modestly across both tasks, which makes it a small but reliable improvement. SOAP underperforms despite extensive tuning, suggesting that preconditioned optimizers with periodic eigenbasis recomputation may not be well suited to the short fine tuning regime. Muon is worth trying as it provided a small consistent gain, while SOAP adds complexity without benefit in our setting.

There are a few ways that we would want to extend this project. Using larger GPT-2 variants would test whether our findings hold across model scales. Constructing higher quality preference pairs by using different kinds of scoring mechanisms (humans vs language models etc) could improve DPO for sonnet generation even further. This could be further extended into multi-step online RL, where we train the base model for multiple steps on new, on-policy data. It would also be interesting to investigate whether different preference pair construction strategies for paraphrase detection could make DPO more effective for classification tasks as well.

Limitations. We only consider GPT-2 base, and results may differ for larger models. Our hyperparameter search, while substantial, is not exhaustive, and a more extensive search could potentially improve SOAP’s results or find a DPO configuration that helps paraphrase detection. The preference pairs for DPO are constructed heuristically rather than from human judgments, which can limit the quality of the preference signal. Our sonnet evaluation metrics are proxies. Human evaluation would be more informative in this case.

8 Conclusion

We investigated optimizer selection and Direct Preference Optimization for improving GPT-2 fine tuning on paraphrase detection and sonnet generation. Our results show that DPO improves sonnet generation (chrF 41.858 vs 39.284) but actually hurts paraphrase detection accuracy (0.875 vs 0.895), revealing that preference optimization is not universally beneficial across task types. For the optimizers, Muon slightly outperforms AdamW (0.897 vs 0.895 paraphrase, 39.928 vs 39.284 chrF), while SOAP slightly underperforms (0.882, 38.892) despite extensive hyperparameter tuning.

The broader takeaway is that the effectiveness of training objectives like DPO depends on the task. For open ended generation where preference pairs capture quality differences that cross entropy cannot, DPO is valuable. For classification tasks where SFT already provides a direct training signal, DPO may not help and can even degrade performance. Our optimizer comparison suggests that Muon’s orthogonalized updates offer a modest but reliable gain in fine tuning, while preconditioned methods like SOAP may not transfer straightforwardly to the short fine tuning regime. Promising

directions can include scaling to larger models, using LLM based preference scoring for generation tasks, and exploring alternative preference pair construction strategies that could make DPO effective for classification.

Key Information

External collaborators: No

External mentor: No

Sharing project: No

Generative AI usage: We used generative AI for brainstorming ideas for further research, along with literature review.

Team Contributions

Arun Moorthy: Implemented the base GPT-2 model, causal self attention, and GPT-2 layer forward pass. Trained sentiment classification baselines. Wrote the introduction and related work sections of the report.

Welton Wang: Implemented SOAP and Muon optimizers. Ran optimizer ablation experiments and hyperparameter tuning for all three optimizers. Wrote the approach and analysis sections of the report.

Andrew Park: Implemented the DPO training pipeline for both paraphrase detection and sonnet generation, including preference pair construction. Ran DPO experiments and hyperparameter ablations. Wrote the experiments and discussion sections of the report.

Late days

Arun Moorthy: 4 late days remaining.

Welton Wang: 5 late days remaining.

Andrew Park: 4 late days remaining.

So, we have 13 total late days with 3 members, meaning we can use 3 late days, which is what we used.

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] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2019.
- [3] Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janber, and Sham Kakade. SOAP: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024.
- [4] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training. *arXiv preprint*, 2025.
- [5] 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. *arXiv preprint arXiv:2305.18290*, 2023.

- [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. *arXiv preprint arXiv:2106.09685*, 2022.
- [7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2015.
- [8] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.
- [9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [10] Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395. Association for Computational Linguistics, 2015.