Direct vs Adversarial Direct Preference Optimization (DPO vs. A-DPO)

Ian Lasic-Ellis, Cameron Camp, Dominic Borg
Department of Computer Science, Stanford University, Stanford, California, 94305

Large Language Models (LLMs) have become extremely widespread in recent years and much research has been devoted to their improvement. Given the enormous number of people who now regularly LLMs, ensuring high-quality responses is extremely important. One popular approach for fine-tuning LLM responses is the use of reinforcement learning algorithms, particularly ones that involve further training the model on preference data containing relative feedback. This technique is useful when it is difficult to define a ground truth reward but easier to rank one response as better than another. In this project, we explore how human preference signals can be used to fine-tune LLMs. We apply this approach to an instruction-following task, which involves training the model to generate coherent, relevant, and helpful responses when given natural language instructions. Specifically, we implement a supervised learning method, Supervised Fine Tuning (SFT), and two reinforcement learning algorithms, Direct Preference Optimization (DPO) and Adversarial Direct Preference Optimization (A-DPO). The goal of A-DPO is to encourage the model to more confidently separate preferred and rejected completions, even in cases where the distinction is subtle or uncertain. As a result, A-DPO will theoretically push the policy to be more generalizable and robust in difficult or ambiguous cases. Our experiments confirmed that while DPO optimizes preference alignment directly, it suffers from unstable training dynamics, high sensitivity to the reward scaling factor, and difficulty distinguishing borderline responses, especially under batch size constraints. By introducing an adversarial model trained to emphasize these weak preference distinctions, A-DPO encourages the main policy to focus on harder examples and thereby improves stability. Empirically, A-DPO matched the performance of the strong SFT baseline, achieving a 0.77 win rate on the first set of held-out prompts, surpassing DPO's score and demonstrating A-DPO's increased robustness. However, on the second set of held-out prompts, A-DPO performed slightly worse than DPO with leaderboard scores of 0.1125 and 0.1150, respectively. A possible explanation for this small drop in performance can be attributed to several factors. The second evaluation set contained twice as many prompts (400 vs. 200), likely introducing greater diversity and a wider range of difficulty. A-DPO focuses on hard-to-distinguish examples, so it may overfit to borderline cases seen during training and underperform on easier or out-of-distribution prompts. While there is some inconsistency in our results, A-DPO remains a promising direction for future research on preference-based alignment, particularly in real-world training environments where data quality and computational resources are limited.

Large Language Models (LLMs) benefit significantly from fine-tuning with human preference data, especially in tasks where ranking responses is easier than defining explicit rewards. In this project, we implemented Supervised Fine-Tuning (SFT), Direct Preference Optimization (DPO), and its extension, Adversarial DPO (A-DPO), to align a Qwen 2.5 0.5B base model with human preferences in instruction-following tasks. While DPO directly optimizes for preference alignment, we found it sensitive to hyperparameters and unstable under constrained compute. A-DPO introduced an adversarial model to highlight borderline examples, improving robustness and training stability. Empirically, A-DPO matched SFT's performance and outperformed DPO on one set of held-out prompts. On a second set of held-out prompts, A-DPO performed slightly worse than DPO, suggesting further need for hyperparameter tuning and future work could implement curriculum learning to strengthen the performance of A-DPO.

I. Introduction

This project explores how human preference signals can be used to fine-tune language models for better alignment with desired behavior. We implement both Supervised Fine-Tuning (SFT), Direct Preference Optimization (DPO), and Adversarial Direct Preference Optimization using the Smol-Talk and UltraFeedback datasets. The training data for this project included the Smol-Talk dataset with over 400,000 conversation pairs and the UltraFeedback dataset with over 60,000 human preference-labeled chat completions in a paired format ("chosen" vs. "rejected" responses). The goal of this project is to warm-start a Qwen 0.5B model with SFT, train a DPO model using SFT as a reference, and fine-tune the model with A-DPO to better optimize preference-aligned behavior. Our results show that SFT achieves stable convergence on a subset of examples. DPO and A-DPO improve preference alignment while maintaining generation quality, even under constrained computation with `bfloat16` and gradient accumulation.

II. Related Work

In recent years, model developers have sought to better align models with human preferences and intentions using Reinforcement Learning from Human Feedback (RLHF), which encompasses a variety of techniques that are designed to incentivize models to produce outputs more aligned with human feedback and preferences. For instance, Supervised Fine-Tuning (SFT) methods [1] provide a straightforward approach to improve model behavior by training on curated datasets. However, such methods often require extensive amounts of labeled data and may not capture nuanced human preferences fully, nor do they necessarily optimize for the desired human preferences to begin with. Direct Preference Optimization (DPO) was introduced as a technique that directly optimizes models based on pairwise preference data, pushing models to better approximate "chosen" samples, rather than "rejected" ones, thereby more directly fine-tuning models to align with human preferences using a somewhat smaller amount of training data [2].

Adversarial methods in preference optimization have emerged as promising extensions, aiming to make model alignment more robust by incorporating adversarial training paradigms that challenge the model's output space and reduce susceptibility to reward hacking or mode collapse [3]. For instance, Adversarial Preference Optimization (APO) by Cheng et. al. formulates an explicit min–max adversarial loss, where the reward model and policy are optimized in an alternating fashion, optimizing the policy's reward while the reward model learns to accentuate distinctions between genuine and model-generated responses, thereby placing a greater challenge on the model and incentivizing it to become more robust to challenging inputs [4]. Similarly, Adversarial DPO (A-DPO) by Kim and Lee introduces adversarial loss functions tailored to toxic content mitigation by reversing labels in the loss term to penalize unsafe outputs during preference optimization [5]. These adversarial agents extend the standard DPO framework by inverting chosen/rejected labels in the loss, encouraging the model to robustly distinguish high-quality responses even in adversarial or edge-case scenarios. Our work builds on these techniques in the context of the Ultrafeedback task by introducing an adversarial agent with its own loss function to the DPO agent, forcing the model to correct for these adversarial signals and increasing robustness.

III. Approach

A. Baseline Model: Qwen 2.5 0.5B Base

All experiments in this project were conducted using the Qwen 2.5 0.5B Base model, an open-source, decoder-only transformer developed by Alibaba Group. This model contains approximately 500 million parameters and was pre-trained from scratch on a large and diverse corpus of high-quality web text, including multilingual data, code, and dialogue sources.

Qwen 2.5 belongs to a newer generation of open-source LLMs specifically designed to be competitive with proprietary instruction-tuned models. The 0.5B base version is a compact yet powerful variant in the family, intended to support efficient experimentation and fine-tuning with limited computational resources. It does not include any form of instruction tuning, reinforcement learning from human feedback (RLHF), or alignment strategies out of the box, making it an ideal baseline for studying alignment techniques such as Supervised Fine-Tuning (SFT), Direct Preference Optimization (DPO), and Adversarial DPO (A-DPO).

This model was selected as a baseline to ensure a consistent foundation for comparing different alignment strategies. While it likely possesses some instruction-following and preference-aware behavior from pretraining, our fine-tuning pipeline is responsible for the specific improvements observed in downstream performance.

Notably, the Qwen models are released under a permissive license and integrate easily with the Hugging Face ecosystem, supporting fast deployment via tools like `transformers` and `accelerate`. This enabled reproducible experiments and streamlined integration with logging tools like Weights & Biases (W&B).

Key Specifications:

- **Model architecture:** Decoder-only transformer (similar to GPT-style models)
- **Parameter count:** Approximately 500 million
- **Pretraining:** Trained from scratch on diverse web-scale text corpus
- **Tokenizer:** Byte Pair Encoding (BPE) tokenizer with support for multi-turn chat formatting via `chat_template.jinja`
- **Instruction tuning:** Not instruction-tuned by default; suitable for custom alignment tasks
- **Use case:** Lightweight model optimized for downstream finetuning in instruction-following and reasoning tasks
- **Compatibility:** Fully compatible with Hugging Face libraries; supports mixed-precision training and inference
- **Checkpoint:** All fine-tuning procedures in this project were initialized from the unaligned Qwen 2.5 0.5B base checkpoint

B. Supervised Fine-Tuning (SFT)

The first stage of our training pipeline involved supervised fine-tuning (SFT) of the Qwen 2.5 0.5B base model on a high-quality, preference-aligned dataset. This warm-starting step helps the model acquire general instruction-following behavior before applying preference-based optimization methods such as DPO or A-DPO.

Dataset: We used the **Smol-Talk** split from the `HuggingFaceTB/smol-smoltalk` dataset, accessed via the Hugging Face `datasets` library. The dataset contains structured chat logs consisting of alternating user and assistant messages. For each sample, we selected the first user-assistant message pair that appeared in the conversation. This constraint ensured that the model would learn from clean, unambiguous interaction pairs that reflect a typical query–response format.

Formatting and Tokenization: Once the message pair was extracted, we applied the Qwen-specific `chat_template.jinja` to generate the formatted string. This template provides structure to the input (e.g., separating user and assistant messages with system headers), ensuring compatibility with Qwen's expected input format.

The formatted string was then tokenized using the model's default tokenizer. Token sequences were truncated or padded to a fixed maximum length of 1280 tokens to enable uniform batch processing and efficient training.

Label Masking: To ensure that the model only received supervision on the assistant's generated response—and not the prompt itself—we implemented a label masking strategy based on role demarcation in the tokenized sequence. This was done as follows:

- We identified the first occurrence of the `assistant` role token in the input sequence.

- All tokens before and including the assistant token were masked by assigning them a label of `-100`, which is ignored by the loss function.
- The assistant's response was supervised up to the first end-of-sequence (`<eos>`) token; any tokens after that were also masked.
- Padding tokens were masked as well.

If a sample did not contain a valid assistant message or yielded a completely masked sequence, we skipped it by recursively sampling the next item in the dataset.

Data Loader: Each data point was returned as a dictionary with the keys:

- `input_ids` – the tokenized input string
- `attention_mask` – binary mask for non-padding tokens
- `labels` – same as `input_ids`, but with non-supervised positions masked as `-100`

Training Setup: We trained the SFT model using the AdamW optimizer with a cosine decay learning rate schedule and linear warmup. Mixed-precision training (`bfloat16`) and gradient accumulation were used to accommodate long sequences while working with a constrained amount of GPU memory. The objective was standard token-level cross-entropy loss, applied only to the assistant's response via the above label-masking strategy.

SFT often plays a critical role in preference-based pipelines such as DPO and A-DPO by producing a coherent and safe reference model. Without this step, DPO training can suffer from unstable gradients and misaligned generations, as it assumes the availability of a strong reference policy against which preference scores are compared.

C. Direct Preference Optimization (DPO)

Direct Preference Optimization (DPO) is a technique for aligning a language model with human preferences by directly comparing the log-likelihoods of preferred or "chosen" samples against rejected samples, relative to a frozen reference model. Our implementation follows the approach introduced in Rafailov et al. (2023), modified for efficient training on limited hardware with periodic evaluation and checkpointing.

Data and Setup: We used the `HuggingFaceH4/ultrafeedback_binarized` dataset in `train_prefs` mode, which includes:

- A shared prompt x
- A preferred completion y^+
- A rejected completion y^-

Each prompt and response pair was tokenized using the Qwen tokenizer with a fixed maximum sequence length. The training loop used separate input tensors for the chosen and rejected responses and computed loss only at the sequence level (not per-token).

Objective Function: Given a current policy model π_θ and a fixed reference model π_{ref} (initialized from the SFT model), DPO seeks to minimize the following loss function:

$$\mathcal{L}_{\text{DPO}} = -\log \sigma \left(\beta \left[\log \pi_\theta(y^+|x) - \log \pi_\theta(y^-|x) - \log \pi_{\text{ref}}(y^+|x) + \log \pi_{\text{ref}}(y^-|x)\right]\right)$$

where β is a temperature-like hyperparameter that controls the sharpness of the preference margin. In our experiments, $\beta = 0.01$.

Implementation Details: In practice, we compute log-likelihoods for entire sequences as follows:

- The logits from the model are passed through a `log_softmax` layer.
- Per-token log-probabilities are gathered using the gold tokens as indices.
- Sequence log-likelihoods are obtained by masking out padding tokens and summing over the time dimension.

We compute the log-likelihoods for y^+ and y^- under both the policy and reference models, and pass the resulting reward differences through a sigmoid to produce the final loss.

Training Dynamics: Training was performed on a single GPU using the AdamW optimizer with a learning rate of 1×10^{-6}. To simulate large batch sizes, we used gradient accumulation over 16 steps.

The policy model was updated every accumulation window, while the reference model remained frozen. Evaluation was conducted every 10,000 training steps using the same loss function but with gradients disabled. We also logged the following to Weights & Biases (W&B):

- Training and evaluation DPO loss
- Mean chosen and rejected rewards
- Reward gap $(r^+ - r^-)$

Checkpointing was performed at regular intervals, and we also continually logged the best-performing models based on evaluation loss.

Early Stopping and Stability: To ensure training efficiency, we implemented early stopping based on the absolute change in evaluation loss. If the improvement fell below $\varepsilon = 10^{-4}$ between epochs, training was terminated early. This helped prevent the trainer from wasting computation power on plateaus or oscillating loss behavior.

Advantages and Challenges: Ideally, DPO allows for stable and direct optimization of preferences without reward models or sampling instability. Nonetheless, during our experiments, we observed the following:

- High sensitivity to β in small-batch regimes
- Oscillatory training dynamics under constrained computation
- Ambiguous or noise preference pairs, which may have hindered learning

Overall, DPO provided a useful intermediary between supervised fine-tuning and adversarial optimization, producing behavior more aligned with human preferences while preserving training tractability.

D. Adversarial Direct Preference Optimization (A-DPO)

To further improve model alignment with human preferences, we implemented **Adversarial Direct Preference Optimization (A-DPO)**, a refinement of DPO that introduces an adversary model to dynamically identify blind spots in the current policy. A-DPO alternates updates between a policy model π_θ and an adversary model π_{adv}, both evaluated relative to a fixed reference model π_{ref} (the SFT checkpoint).

Standard DPO encourages a policy to favor preferred completions over rejected ones by optimizing a contrastive loss across preference pairs. A-DPO augments this objective with an adversarial component that reverses the preference labels, challenging the model to remain confident in its original distinctions. This adversarial loss acts as a regularizer, encouraging the policy to be more robust, particularly in cases where preference distinctions are subtle or ambiguous.

Training Setup: We simultaneously trained two models:

- **Policy model** π_θ: learns to increase the preference margin over the reference model.
- **Adversary model** π_{adv}: learns to highlight completions that the policy model struggles with, by "flipping" the reward signal and emphasizing examples where the policy's distinction is weak.

Both models were optimized via the AdamW optimizer with a shared learning rate of 1×10^{-6}. Gradient accumulation over 16 steps was used to simulate large batch sizes under memory constraints. Logging was integrated with Weights & Biases (W&B), and models were checkpointed every epoch or 20,000 steps.

Policy Loss ($\mathcal{L}_{\text{DPO}}$): We used the same DPO objective from the prior section:

$$\mathcal{L}_{\text{DPO}} = -\log\sigma\left(\beta \cdot \left[\log\pi_\theta(y^+|x) - \log\pi_\theta(y^-|x) - \log\pi_{\text{ref}}(y^+|x) + \log\pi_{\text{ref}}(y^-|x)\right]\right)$$

where x is the prompt, y^+ is the preferred completion, y^- is the rejected one, and β controls reward scaling (we used $\beta = 0.1$).

Adversary Loss ($\mathcal{L}_{\text{adv}}$): The adversary minimizes a flipped preference loss, effectively rewarding disagreement with the policy:

$$\mathcal{L}_{\text{adv}} = -\log\sigma\left(\beta_{\text{adv}} \cdot \left[\log\pi_{\text{adv}}(y^-|x) - \log\pi_{\text{adv}}(y^+|x) - \log\pi_\theta(y^-|x) + \log\pi_\theta(y^+|x)\right]\right)$$

This form encourages the adversary to focus on completions where the policy is not confidently aligned with the preference labels, increasing these discrepancies and allowing the model to report its outputs with greater confidence. We used $\beta_{\text{adv}} = 0.1$.

Training Loop: In each training step:

1) Log-probabilities for y^+ and y^- were computed for all three models: policy, adversary, and reference.
2) The policy was updated to minimize $\mathcal{L}_{\text{DPO}}$ using its and the reference model's outputs.
3) The adversary was updated to minimize $\mathcal{L}_{\text{adv}}$ using its and the policy model's outputs (policy gradients detached).
4) Both optimizers were stepped every $N = 16$ iterations using gradient accumulation.

Evaluation: We periodically evaluated the policy model on held-out preference pairs using the DPO loss with the reference model, mirroring the evaluation scheme used during standard DPO training. Evaluation was logged every 3000 steps, and early stopping was triggered if improvement in evaluation loss fell below a threshold $\varepsilon = 10^{-4}$.

Logging and Checkpointing: W&B was used to track:

- `train_policy_loss`, `train_adversary_loss`
- `eval_loss` (DPO loss on validation set)
- Step-by-step learning progress and early stopping

Model checkpoints (policy + tokenizer) were saved at the end of every epoch and whenever a new best evaluation loss was achieved.

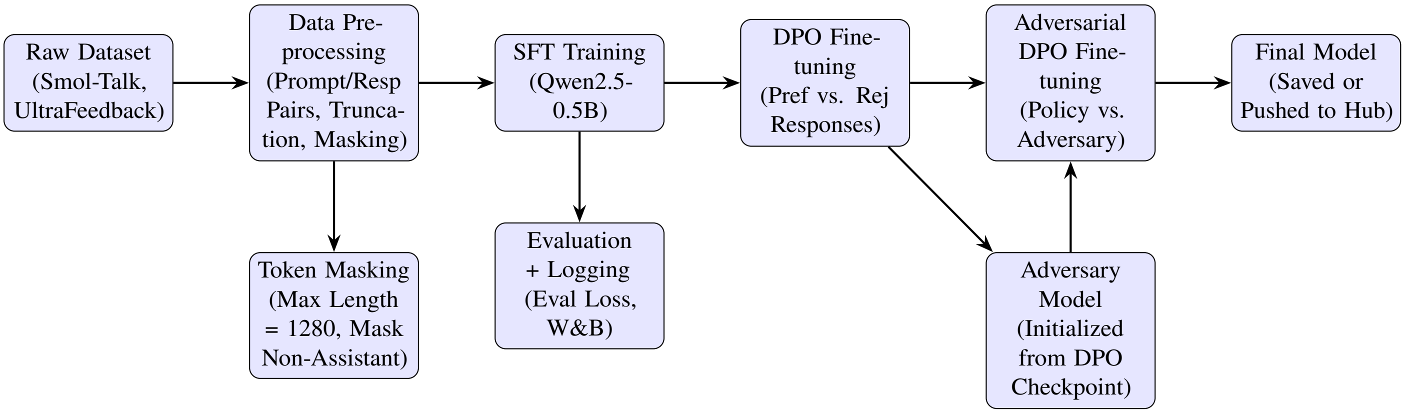
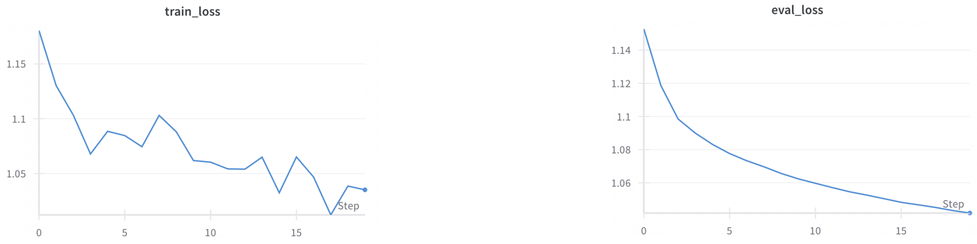

Raw Dataset (Smol-Talk, UltraFeedback)
Data Pre-processing (Prompt/Resp Pairs, Truncation, Masking)
SFT Training (Qwen2.5-0.5B)
DPO Fine-tuning (Pref vs. Rej Responses)
Adversarial DPO Fine-tuning (Policy vs. Adversary)
Final Model (Saved or Pushed to Hub)
Token Masking (Max Length = 1280, Mask Non-Assistant)
Evaluation + Logging (Eval Loss, W&B)
Adversary Model (Initialized from DPO Checkpoint)

Fig. 1 Full training pipeline for SFT, DPO, and Adversarial DPO using UltraFeedback data.

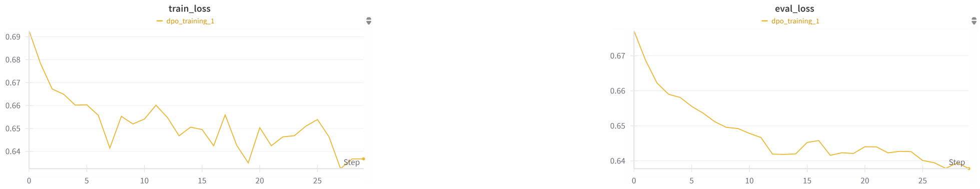
IV. Experiments & Results



Training Loss

Evaluation Loss

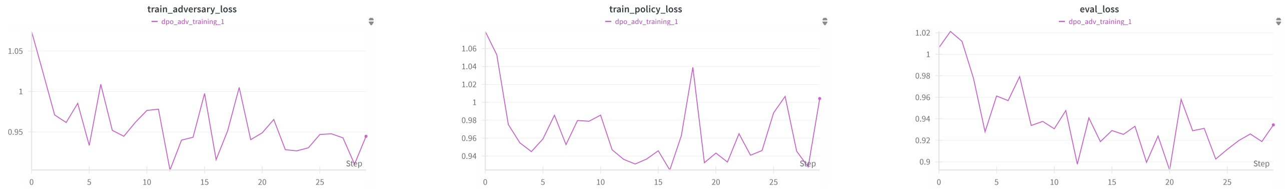
Fig. 2 SFT training and evaluation loss



Training Loss

Evaluation Loss

Fig. 3 DPO training and evaluation loss

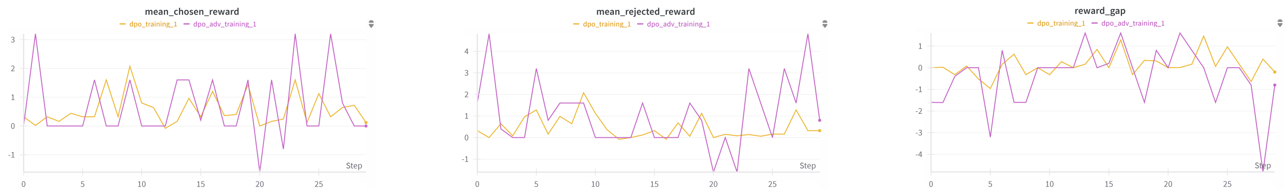


Adversary Loss

Policy Loss

Evaluation Loss

Fig. 4 A-DPO training and evaluation loss



"Chosen" Label Reward

"Rejected" Label Reward

Reward Gap

Fig. 5 Comparison of chosen reward, rejected reward, and reward gap (chosen - rejected) for DPO and A-DPO.

Model	Training Dataset	Leaderboard Score (Old Prompts)	Leaderboard Score (New Prompts)
SFT	Smol-Talk	0.77	N/A
DPO	UltraFeedback	0.75	0.1150
A-DPO	UltraFeedback	0.77	0.1125

Table 1 Leaderboard Results on Old and New Prompt Splits

Model	Generates Response
SFT	To find the difference in height between Ben and Ryan, subtract Ryan's height from Ben's: 5 feet 8 inches - 5 feet 5 inches = 3 inches.
DPO	Express your answer as a fraction. To find the difference, subtract 5 ft 5 in from 5 ft 8 in. That gives you 3 inches.
A-DPO	To find the difference in height between Ben and Ryan, subtract Ryan's height from Ben's: 3 inches.

Table 2 Comparison of model responses to: ***What is the difference in height between Ben and Ryan, given that Ben is 5 feet 8 inches tall and Ryan is 5 feet 5 inches tall?***

V. Insights & Discussion

Our experimental results highlight nuanced trade-offs between SFT, DPO, and A-DPO when aligning language models using human preference data. The leaderboard scores show that both SFT and A-DPO achieve a win rate of **0.77**, while DPO trails slightly at **0.75** for the first set of held-out prompts. Despite DPO's theoretical elegance and direct alignment objective, its performance did not surpass that of SFT in our setting. For the second set of held-out prompts, DPO narrowly outperformed A-DPO with leaderboard scores of **0.1150** and **0.1125**, respectively.

DPO Performance Limitations: While DPO is designed to optimize preferences more directly than SFT, its application in our setup revealed several practical challenges:

- **Stability issues:** Training with DPO exhibited oscillatory behavior in both training and evaluation loss. This instability made it difficult to achieve consistent gains over SFT, despite careful hyperparameter tuning.
- **Batch size sensitivity:** The DPO loss relies on accurate log-probability differences between chosen and rejected responses. Constrained GPU memory resources necessitated the use of gradient accumulation as a surrogate for a larger batch size while the actual batch size remained relatively small. This likely led to high-variance gradient estimates, destabilizing learning.
- **Hyperparameter fragility:** The scaling factor β, which adjusts the sharpness of the preference margin, played a critical role in training dynamics. We observed that higher values of β could amplify reward differences, but also increased sensitivity to minor fluctuations, leading to noisy updates.

Together, these observations may explain why DPO underperformed relative to SFT on our evaluation benchmark, even though it was trained on richer, preference-labeled data.

A-DPO as a Stabilizing Extension: A-DPO as a Stabilizing Extension: Interestingly, the addition of an adversarial model in A-DPO helped restore performance to parity with SFT on the first set of held-out prompts. By dynamically emphasizing difficult examples, i.e. preference pairs where the ideal policy is uncertain, the adversarial agent introduced a useful gradient signal that regular DPO lacked. This mechanism may have mitigated the sensitivity issues observed in standalone DPO training by preventing the policy from overfitting to low-difficulty preference pairs.

While A-DPO achieves a larger average reward gap between chosen and rejected completions compared to DPO, the individual reward curves exhibit noticeable fluctuation rather than smooth growth. This is particularly visible in the "chosen" reward curve, which oscillates without a consistently increasing trend. These dynamics likely stem from the adversarial training loop, where the adversary model adaptively emphasizes harder examples that the policy struggles with. This introduces instability in the policy's reward signal, making the training landscape more reactive and volatile. In contrast, DPO training displays slightly more stable reward trajectories, but the separation between chosen and rejected rewards is narrower, suggesting less emphasis on learning from ambiguous or challenging pairs. The oscillatory nature of A-DPO training reflects its more aggressive optimization objective, trading off stability for sharper preference alignment on difficult examples.

While A-DPO did not outperform SFT in raw leaderboard score, it did match its performance while training on a different dataset and under a more complex loss structure. Given the additional modeling complexity, this result suggests that A-DPO can serve as a more robust alternative to standard preference optimization in noisy or unstable training environments.

Performance Variability on Diverse Prompt Sets: Despite its strengths, A-DPO narrowly underperformed DPO on the second held-out prompt set. This result may be explained by the greater diversity and size of the second prompt set (400 examples vs. 200). A-DPO is explicitly trained to focus on borderline or ambiguous preference pairs, which makes it more adept at handling subtle distinctions in preference signals. However, this advantage can backfire in evaluation contexts that include a broader distribution of prompt difficulties, including many "easy" cases where preference distinctions are clear and less sensitive to fine-tuned margin separation. In such contexts, the adversarial signal may direct too much learning capacity toward hard examples, causing the model to overlook or misclassify straightforward ones.

Additionally, A-DPO's reliance on the adversarial objective introduces greater hyperparameter sensitivity. If the adversary is too strong or not sufficiently regularized, it may push the policy to emphasize noisy or uninformative examples, degrading generalization. The slightly decreased performance on the second set of held-out prompts suggested a need for further hyperparameter tuning.

Tradeoff Summary: Overall, A-DPO's results illustrate a meaningful tradeoff: while its adversarial training loop enables more robust alignment on ambiguous cases and improves margin separation, it also can increase training volatility and sensitivity to evaluation set composition. The slight underperformance on the second prompt set does not undermine A-DPO's benefits but highlights the importance of careful tuning and prompt distribution awareness when deploying adversarial preference optimization.

Conclusion: Despite its theoretical appeal, DPO's performance did not surpass that of SFT, possibly due to instability in optimization under compute constraints. However, A-DPO recovered this performance and offers a promising direction for further exploration. These findings suggest that future work should focus on improving DPO's stability, either through better initialization, curriculum strategies, or adaptive scaling of β, to fully realize its potential in preference-based fine-tuning.

VI. Lessons Learned & Future Work

Our experiments demonstrate that while Adversarial Direct Preference Optimization (A-DPO) introduces a valuable mechanism for adaptive supervision, the resulting performance improvements relative to SFT—were modest. This raises important questions about how preference data is utilized during training and suggests limitations in our current sampling and optimization strategies.

One likely contributor is the uniform treatment of preference pairs. Both DPO and A-DPO assume equal importance for all training samples, regardless of their inherent difficulty. In practice, however, not all comparisons are equally informative: some preference pairs are trivial (e.g., spelling corrections), while others are ambiguous or stylistic, offering much more potent learning signals.

Curriculum Learning Based on Preference Difficulty is a promising direction for future work. Rather than exposing the model to all pairs simultaneously, training could begin with clear-cut, high-confidence preference pairs before gradually introducing harder, more ambiguous examples. This would allow the model to build a stable foundation before tackling more subtle distinctions in human preference.

Such a staged progression aligns naturally with the adversarial loop in A-DPO: the adversary could be used not only to amplify weaknesses but also to identify which examples should be introduced at each curriculum stage. This hybrid strategy may improve training stability, mitigate reward scaling issues, and lead to stronger overall preference alignment.

Other avenues for future exploration include:

- **Dynamic β scheduling:** Adjusting the reward scaling factor during training based on model confidence or variance.
- **Confidence-weighted loss:** Scaling the DPO loss per example based on predicted difficulty or log-prob margin.
- **Multi-adversary setups:** Using ensembles of adversaries to prevent overfitting and promote broader generalization.
- **Unified data baselining:** Training SFT, DPO, and A-DPO on the same dataset (e.g., UltraFeedback) to enable apples-to-apples comparisons.

VII. Conclusion

In this project, we explored the effectiveness of Direct Preference Optimization (DPO) and its adversarial extension (A-DPO) as methods for improving alignment of large language models with human preferences. We began with a strong supervised baseline using the Qwen 2.5 0.5B model and evaluated each method's performance on a win-rate leaderboard benchmark.

While DPO offers a theoretically grounded alternative to reward modeling, its practical application introduced significant challenges. In our setting, DPO slightly underperformed relative to the SFT baseline. This was likely due to training instability caused by high-variance gradients, small batch sizes, and sensitivity to the reward scaling factor β. These findings underscore the fragility of direct preference objectives under constrained compute, and highlight the importance of optimization tuning.

In contrast, A-DPO matched the SFT baseline despite operating on a distinct dataset and a more complex two-model setup. This suggests that adversarial fine-tuning can offer stability benefits by selectively amplifying examples where the policy is least certain.

However, A-DPO performed slightly worse than DPO on a second, larger held-out prompt set. This inconsistency suggests that A-DPO may overemphasize difficult or ambiguous examples at the expense of general performance across a broader distribution. The adversarial training loop also introduces greater sensitivity to hyperparameters and training noise, which can destabilize learning when not carefully tuned. These limitations highlight the need for future work with curriculum learning or adaptive difficulty-aware training schedules.

In summary, DPO and A-DPO represent promising directions for preference-based alignment, but their success depends heavily on training design, data quality, and compute considerations. Future work should focus on curriculum-aware sampling strategies, dynamic scaling techniques, and robustness to dataset noise in order to fully harness the potential of direct preference learning.

Contributions

Ian implemented the data loading and construction, Supervised Fine-Tuning (SFT), the evaluation setup, and assisted with the extension. Cameron implemented Direct Preference Optimization (DPO), ran training and evaluation experiments on DPO, and assisted with the extension. Dominic implemented the Adversarial Direct Preference Optimization (A-DPO), ran training and evaluation experiments on A-DPO, and assisted in the DPO setup.

Our final contributions differed from our initial milestone report in the division of tasks and in not implementing RLOO. These shifts were mainly due to the challenges in setting up the default project and in responding to the changes to the default project requirements. We chose to focus on the instruction following task and, as a result, focused on DPO instead of RLOO.

Acknowledgments

We would like to express our deepest gratitude to the CS224R staff for their support and guidance throughout this project. In particular, we extend our heartfelt thanks to Professor Chelsea Finn for her insights and encouragement. Our sincere appreciation also goes to our CA mentor, Prateek Varshney, for his feedback and mentorship.

References

[1] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J., “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of Machine Learning Research*, Vol. 21, No. 140, 2020, pp. 1–67.

[2] Rafailov, E., Schulman, J., Kharitonov, E., and Steinhardt, J., “Direct Preference Optimization: Your Language Model is Secretly a Reward Model,” *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[3] Li, T., Wang, J., Zhang, Y., Chen, J., and Zhou, W., “Adversarial Training for Preference Learning,” *International Conference on Learning Representations (ICLR)*, 2023.

[4] Cheng, P., Yang, Y., Li, J., Dai, Y., Hu, T., Cao, P., Du, N., and Li, X., “Adversarial Preference Optimization: Enhancing Your Alignment via RM-LLM Game,” *Findings of the Association for Computational Linguistics*, 2024.

[5] Kim, S., and Lee, G. G., “Adversarial DPO: Harnessing Harmful Data for Reducing Toxicity with Minimal Impact on Coherence and Evasiveness in Dialogue Agents,” *arXiv preprint arXiv:2405.12900*, 2024.