Extended Abstract

Motivation Recent advances in large language models (LLMs) have demonstrated that reinforcement learning from human feedback (RLHF) can significantly enhance alignment and usability. However, traditional methods like PPO are computationally expensive and require a separate reward model. Emerging preference-based methods—such as Direct Preference Optimization (DPO), Odds Ratio Preference Optimization (ORPO), and Tapered Off-Policy REINFORCE (TOPR) offer more efficient alternatives. Our goal is to systematically evaluate and compare several of these methods in terms of effectiveness, training efficiency, and robustness.

Method We compare five fine-tuning approaches: DPO, ORPO, TOPR, KTO, and RLOO. Each method uses pairwise or binary preference data to improve a base instruction-tuned model. DPO optimizes a log-ratio objective based on human preferences. ORPO augments supervised fine-tuning (SFT) with a contrastive odds ratio term, eliminating the need for a reference model. TOPR applies asymmetric importance sampling to efficiently update from both positive and negative examples. KTO incorporates loss aversion from Prospect Theory using binary reward signals. RLOO introduces variance reduction via leave-one-out baselines in the REINFORCE gradient.

Implementation We use Qwen/Qwen2.5-0.5B as the base model, initially fine-tuning it on 460K examples from the SmolTalk dataset (SFT). Reinforcement-style preference tuning is then performed using 67K chosen-rejected pairs from the UltraFeedback Binarized dataset. Hyperparameters, such as learning rate, batch size, and regularization terms, are tuned based on method-specific guidance.

Results ORPO emerged as the most effective method. ORPO V1.1, trained on top of the SFT model, achieved a 91.0% win rate against the Qwen base model and 87.4% against our SFT model, securing the highest leaderboard scores. Even ORPO V1.2, trained from scratch without SFT, reached a 70.1% win rate against SFT while using only 20% of the training time. TOPR consistently outperformed DPO across all metrics and delivered strong results without requiring a reward model. DPO V3.1 showed better loss curves with batch size 8, but underperformed in leaderboard evaluations, likely due to a shorter training sequence length of 256. KTO performed competitively despite relying solely on binary feedback. RLOO degraded model performance and was the weakest overall, likely due to suboptimal hyperparameter tuning.

Discussion ORPO's strength lies in its simple and stable objective, which effectively penalizes rejected outputs without needing a reference model. Its flexibility and high sample efficiency make it an excellent choice for alignment-focused fine-tuning. While slightly less performant than ORPO, TOPR proved to be both stable and effective, benefiting from its ability to leverage negative examples. DPO was sensitive to hyperparameters and batch size, and its reliance on log-ratio objectives led to less stable training. KTO presents a scalable alternative with minimal data requirements. RLOO, though promising in theory, requires more rigorous hyperparameter tuning to be effective.

Conclusion We present a systematic comparison of recent preference-based RL methods for language model fine-tuning. Among the tested approaches, ORPO demonstrated the best balance of performance, stability, and implementation simplicity. It achieves high alignment without a reference model and can even train effectively from scratch, making it highly flexible and efficient. TOPR also proved effective, particularly in settings with both positive and negative feedback. DPO requires careful tuning and was less robust in practice. KTO offered a strong baseline despite its use of binary feedback. RLOO was the least successful in our setup but may benefit from further tuning. These findings suggest that future work should prioritize methods like ORPO and TOPR for real-world LLM fine-tuning. Improvements such as multi-epoch training, more diverse datasets, and larger-scale models may further enhance performance.

RL Fine-Tuning of Large Language Models

Gerardus de Bruijn
Department of Computer Science
Stanford University
gdebruyn@stanford.edu

Nareauphol Liu
Department of Electrical Engineering
Stanford University
gliu29@stanford.edu

Dev Jayram
Department of Mathematics
Stanford University
devj10@stanford.edu

Abstract

We conduct a comparative study of several recent methods for fine-tuning large language models using human preference data, focusing on DPO, ORPO, TOPR, KTO, and RLOO. Our experiments are conducted on the Qwen2.5-0.5B model using the SmolTalk and UltraFeedback Binarized datasets. We find that ORPO consistently outperforms all other methods, achieving high alignment with minimal training cost, even when initialized from the base model without supervised pre-training. TOPR shows promising results by leveraging both positive and negative feedback, while KTO achieves competitive performance using only binary reward signals. In contrast, DPO proves to be sensitive to hyperparameter tuning and sequence length. Overall, we highlight the trade-offs between stability, simplicity, and performance across methods, and recommend ORPO and TOPR as practical and efficient approaches for preference-based fine-tuning of LLMs.

1 Introduction

Reinforcement learning (RL), particularly REINFORCE-style methods, is central to fine-tuning large language models (LLMs) for complex tasks such as chain-of-thought reasoning. However, traditional REINFORCE suffers from instability and inefficiency in off-policy training settings, especially when leveraging both positive and negative examples. While KL regularization can help stabilize training, it often slows learning and requires careful hyperparameter tuning.

Given the increasing role of LLMs in real-world applications, developing robust and sample-efficient fine-tuning methods is critical. Algorithms that can effectively incorporate both positive and negative feedback can reduce training costs, minimize inference waste, and prevent model degradation—ultimately improving the scalability and reliability of LLM deployment.

To address these challenges, we propose Tapered Off-Policy REINFORCE (TOPR), an algorithm that remains stable even when the model diverges significantly from the data distribution. TOPR efficiently utilizes both positive and negative examples through an asymmetric policy gradient formulation that encourages desired behaviors while discouraging undesired ones.

In addition, we introduce Odds Ratio Preference Optimization (ORPO), a novel approach that simultaneously improves task performance and alignment with human preferences. Unlike DPO and TOPR, ORPO eliminates the need for a reference model, making it more lightweight and flexible.

Stanford CS224R 2025 Final Report

Figure 1: During training, log probabilities of rejected responses increase along with the chosen ones.

While not detailed extensively in this paper, we have also explored preliminary versions of Reinforce Leave-One-Out (RLOO) and Kahneman-Tversky Optimization (KTO), which uses a binary preference signal to indicate whether an output is desirable.

We evaluate and compare the performance and efficiency of these methods, highlighting their trade-offs and practical implications for fine-tuning LLMs.

2 Related Work

2.1 Tapered Off Policy REINFORCE (TOPR)

Roux et al. (2025) introduces TOPR, a reinforcement learning algorithm that leverages both positive and negative examples in an off-policy setting. Traditional RL methods often struggle in this domain due to instability when the model distribution diverges significantly from the training data—particularly in the presence of negative rewards. To address this, TOPR incorporates a truncated importance sampling (TIS) strategy that, fully upweights positive trajectories, downweights negative trajectories in proportion to how likely the model is to generate them, and avoids reliance on Kullback–Leibler (KL) regularization.

2.2 Odds Ratio Preference Optimization (ORPO)

Hong et al. (2024) proposes ORPO, a novel alignment method that unifies SFT and preference alignment into a single stage. ORPO operates directly on the target model, without requiring a separate reference model, by augmenting the standard SFT loss with an additional term that penalizes the log-probability of rejected tokens. This addresses a key limitation of the traditional cross-entropy loss, which tends to increase the log-probabilities of both correct and incorrect tokens during training. As illustrated in Figure 1, this dynamic can unintentionally reinforce undesirable behavior, motivating ORPO's corrective loss term.

2.3 Kahneman-Tversky Optimization (KTO)

Ethayarajh et al. (2024) introduces KTO, a preference optimization approach inspired by Prospect Theory, which models how humans make decisions under uncertainty. Unlike ORPO and DPO, which maximize the log-likelihood of preferred outputs, KTO focuses on maximizing the utility of generations. It explicitly incorporates human loss aversion, assigning greater weight to avoiding negative outcomes. KTO requires only a binary feedback signal indicating whether a response is desirable, making it lightweight and broadly applicable in scenarios where nuanced preference labels are unavailable.

3 Method

We assume access to a dataset of preference pairs where each sample contains a prompt and two candidate responses: a preferred (chosen) response and a rejected one. Formally, the dataset is defined as:

$$\mathcal{D} = \left\{ x^{(i)}, y_w^{(i)}, y_l^{(i)} \right\}_{i=1}^{N}.$$

where $x^{(i)}$ is the prompt, $y_w^{(i)}$ the preferred answer, and $y_l^{(i)}$ the rejected answer.

RLHF algorithms typically consist of two main components: a reward signal aligned with human preferences and a Kullback–Leibler (KL) divergence penalty to prevent the fine-tuned policy π_θ from straying too far from the original model π_{ref}. The general objective can be written as:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim D, y\sim\pi_\theta(y|x)} \left[r_\phi(x, y) \right] - \beta \mathbb{D}_{KL} \left[\pi_\theta(y|x) || \pi_{\text{ref}}(y|x) \right]$$

Here, β is a hyperparameter controlling the strength of the KL constraint. The policy π_θ is initialized from the SFT-trained reference model π_{ref} and fine-tuned using the above objective above.

3.1 Tapered Off Policy REINFORCE (TOPR)

TOPR uses importance sampling to down-weight negative trajectories not likely under π, while allowing positive trajectories to be up-weighted irrespective of π. The gradient can be written as:

$$\nabla J_{\text{TOPR}}(\pi) = \sum_{\tau: R(\tau)\geq 0} \mu(\tau) R(\tau) \nabla \log \pi(\tau) + \sum_{\tau: R(\tau)<0} \mu(\tau) \left[\left(\frac{\pi(\tau)}{\mu(\tau)} \right)^{[0,1]} \right] R(\tau) \nabla \log \pi(\tau)$$

The upper truncation limits allow us to keep gradient variance under control. The lower limit is to allow the algorithm to progressively reduce the contribution of negative trajectories to the gradient. This combines the SFT update for positive examples, leading to acceleration, and the truncated importance sampling update for negative examples to reduce the contribution of negative trajectories to the gradient. This way we are gracefully unlearning negative trajectories.

3.2 Direct Preference Optimization (DPO)

A subjective reward indicating how good a response is can be difficult to collect. In contrast, it is much easier to collect pairwise preferences. These can be annotated by humans or synthetically generated, for example by sampling two different responses from an SFT model with non-zero temperature (i.e., non-deterministic decoding).

According to the Bradley Terry model the probability a given pairwise preference is true can be defined as:

$$p^*(y_w \succ y_l \mid x) = \frac{exp(r^*(x, y_w))}{exp(r^*(x, y_w)) + exp(r^*(x, y_l))}$$

Rafailov et al. (2023) show that, with a few mathematical steps, the DPO objective can be derived without requiring any explicit rewards:

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

This objective maximizes the *difference* in implicit reward between the preferred and rejected responses by increasing the likelihood of the preferred response over the rejected one.

3.3 Odds Ratio Preference Optimization (ORPO)

While DPO eliminates the need for training a separate reward model, Odds Ratio Preference Optimization (ORPO) Hong et al. (2024) goes further by combining fine-tuning and preference alignment

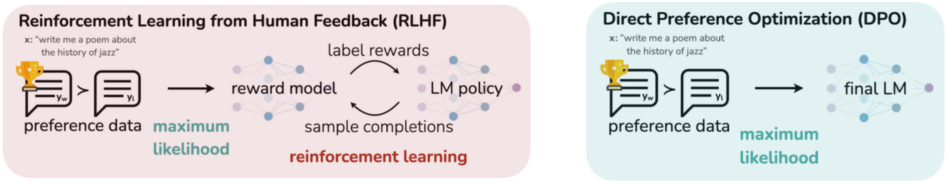

Figure 2: RLHF requires training a separate reward model. DPO maximizes likelihood of chosen while minimizing rejected of policy (model) being trained vs reference policy.

into a single stage. The ORPO objective function consists of the negative log-likelihood loss typically used in the SFT stage plus an additional term for penalizing undesirable outputs. This is achieved by maximizing the odds ratio of the likelihood of generating a desirable output versus an undesirable output:

$$\mathcal{L}_{ORPO} = \mathbb{E}_{(x,y_w,y_l)\sim D}\left[\mathcal{L}_{SFT} + \mathcal{L}_{OR}\right]$$

Here $\mathcal{L}_{OR}$ is defined as:

$$\mathcal{L}_{OR} = -\lambda \log \sigma\Big(\log \frac{\textbf{odds}_\theta(y_w|x)}{\textbf{odds}_\theta(y_l|x)}\Big)$$

where x is the input sequence, y_w is the desirable output sequence, y_l is the undesirable output sequence, and λ is a hyperparameter. The odds of a sequence y given a sequence x is defined as:

$$\textbf{odds}_\theta(y|x) = \frac{P_\theta(y|x)}{1 - P_\theta(y|x)}$$

Since ORPO only requires the target model π_θ, no reference policy is needed. This enables a single-stage training procedure. Figure 1 shows how both good and bad response log-probabilities can increase during SFT. This behavior can be explained by how the cross-entropy loss is formulated:

$$\mathcal{L} = -\frac{1}{m}\sum_{k=1}^{m}\sum_{i=1}^{|V|} y_i^{(k)} \cdot \log(p_i^{(k)})$$

Because $y_i^{(k)}$ is 1 only for the correct token and 0 otherwise, the loss does not penalize high probabilities for incorrect tokens. This explains why the probabilities of rejected responses also tend to increase during SFT. ORPO mitigates this by introducing a contrastive gradient:

$$\nabla_\theta \mathcal{L}_{\text{OR}} = \delta(d) \cdot h(d)$$

For $d = (x, y_w, y_l)$:

$$\delta(d) = \left[1 + \frac{\textbf{odds}_\theta P(y_w|x)}{\textbf{odds}_\theta P(y_l|x)}\right]^{-1}$$

$$h(d) = \frac{\nabla_\theta \log P_\theta(y_w|x)}{1-P_\theta(y_w|x)} - \frac{\nabla_\theta \log P_\theta(y_l|x)}{1-P_\theta(y_l|x)}$$

Here, one term penalizes wrong predictions and the other term contrasts between chosen and rejected responses. $\delta(d)$ will converge to 0 when the odds of the chosen responses are relatively higher than the rejected responses. $h(d)$ implies a weighted contrast of the two gradients from the chosen and rejected responses. Meanwhile, the denominator $1 - P(y|x)$ amplifies gradients when $P(y|x)$ is large, accelerating adaptation toward desired outputs.

3.4 Kahneman-Tversky Optimization (KTO)

Ethayarajh et al. (2024) show that many preference optimization objectives implicitly encode loss aversion biases, as described in Prospect Theory. KTO formalizes this insight by introducing an objective that leverages these biases explicitly.

KTO only requires a binary feedback for a response. The KTO objective is:

$$\mathcal{L}_{KTO}(\pi_\theta; \pi_{\text{ref}}) = \mathbb{E}_{(x,y)\sim D}\left[\lambda_y - v(x,y)\right]$$

$$r_\theta(x,y) = \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)},\ z_0 = \mathbb{E}_{x'\sim D}\left[\mathbb{KL}(\pi_\theta(y'|x')||\pi_{\text{ref}}(y'|x'))\right]$$

$$v(x,y) = \begin{cases} \lambda_D \sigma(\beta(r_\theta(x,y) - z_0)) & if\ y \sim y_{desirable}|x \\ \lambda_U \sigma(\beta((z_0 - r_\theta(x,y))) & if\ y \sim y_{undesriable}|x \end{cases}$$

The objective clearly captures the essence of loss aversion by weighting penalties for bad responses λ_U more heavily than rewards for good ones λ_D, similar to how humans are more sensitive to losses than to gains. An existing preference data set of n chosen / rejected pairs already used for DPO and ORPO training can be split into $2n$ binary labeled examples.

4 Experimental Setup

4.1 Model

We use Qwen/Qwen2.5-0.5 as the base model. This model is based on the decoder-only transformer architecture and is capable of generating text conditioned on input sequences. At 0.5 billion parameters, it is a relatively small instruction-tuned language model. We first apply SFT to this model, then use the resulting SFT checkpoint as the starting point for all subsequent fine-tuning methods we evaluate.

4.2 Datasets

For the initial SFT stage, we use the SmolTalk dataset, which contains approximately 460K instruction-following examples. For fine-tuning with preference-based methods, we use the Huggingface's UltraFeedback Binarized dataset, which provides 67K pairwise comparisons between chosen and rejected model responses.

4.3 Training Setup

We experiment with two different strategies for constructing prompts and labels during SFT:

- **SFT v1.2:** Uses the encoded prompt as input and the corresponding completion as the target label. Training was performed on a single NVIDIA T4G Tensor Core GPU with 16GB memory and 8 vCPUs (AWS `g5.2xlarge` instance).
- **SFT v2.6:** Concatenates the prompt and completion, appending the special end-of-turn token `<|im_end|>` after each turn. This setup was trained on a single NVIDIA L4 Tensor Core GPU with 24GB memory and 8 vCPUs (`g6.2xlarge` instance).

All subsequent reinforcement learning (RL)-based fine-tuning methods (DPO, RLOO, TOPR, KTO) are initialized from the SFT v2.6 checkpoint. In each case, a frozen copy of the SFT model serves as the reference policy (i.e., the behavior policy), while the fine-tuned model is treated as the target policy.

4.4 Training run details

All fine-tuning runs use a single epoch over the UltraFeedback Binarized dataset with a maximum sequence length of 512 tokens. SFT is trained with a batch size of 8, while DPO, RLOO, TOPR, and KTO use a batch size of 4 due to increased memory usage. To explore memory-speed tradeoffs, we also train alternative DPO and KTO models using a smaller max sequence length of 256 and a larger batch size of 8. Additional hyper parameters are listed below:

- **Learning rate:** SFT: $2e^{-5}$; DPO, RLOO and TOPR: $3e^{-6}$; ORPO: $8e^{-6}$; KTO: $1e^{-5}$
- **Learning rate decay:** For all non-SFT methods, we apply a $0.9\times$ decay every 1500 iterations.
- **Gradient accumulation:** Optimizer steps occur every 8 iterations, which results in roughly a 10% speed improvement.

Method	SFT v1.2	SFT v2.6	ORPO	TOPR	DPO	KTO	DPO batch 8	KTO batch 8
	3.91	1.92	1.95	1.73	1.58	1.58	1.57	1.59

Table 1: Iterations per second for each method

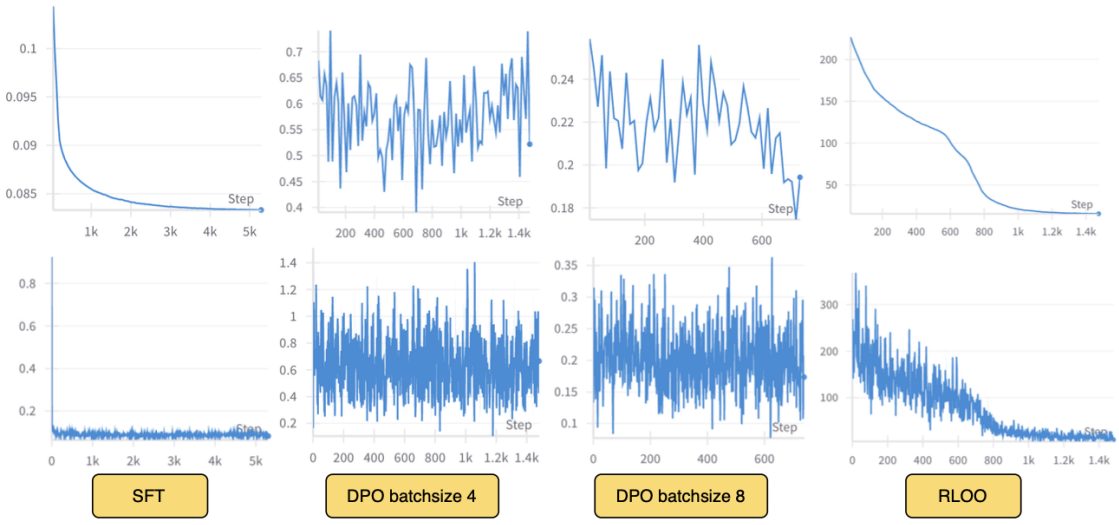

Figure 3: Validation loss (top) and training loss (bottom) during one epoch for SFT and RL methods DPO and RLOO

- **Regularization terms (from original papers):** $\beta_{\text{DPO}} = 0.5$, $\lambda_{\text{ORPO}} = 0.1$, $\gamma_{\text{TOPR}} = 1.0$, and $\beta_{\text{KTO}} = 0.1$.

5 Results

5.1 Quantitative Evaluation

5.1.1 Training results

The training loss and validation loss curves for SFT, DPO and RLOO are depicted in Figure 3. Figure 4 has the loss curves for TOPR, ORPO and KTO. Table 1 lists the speed in iterations per second for each method.

We also implemented LoRA for parameter-efficient fine-tuning (PEFT). On a local laptop, LoRA enabled training with double the batch size during unit tests.

5.1.2 Evaluation

We sampled 1,000 prompts from a held-out split of the UltraFeedback dataset. Each prompt was processed by both our model and a reference model to generate responses. Both models were hosted internally on a VLLM server. Response quality was scored by the externally hosted `llama-3.1-nemotron-70b-reward` model (OpenAI), which returned a scalar reward. The win rate was calculated as the percentage of prompts for which our model's response received a higher reward than the reference model's. All models were trained with a maximum sequence length of 512, and the same constraint was applied during evaluation.

Table 2 lists the fine-tuning methods along with their base models and target model versions. It includes win rates against various reference models, leaderboard scores, and total training time.

Appendix A has the list of all the leaderboard submissions as well as further experiments with finetuning DPO and ORPO where we used a maximum length of 1024 instead of 512. DPO originally

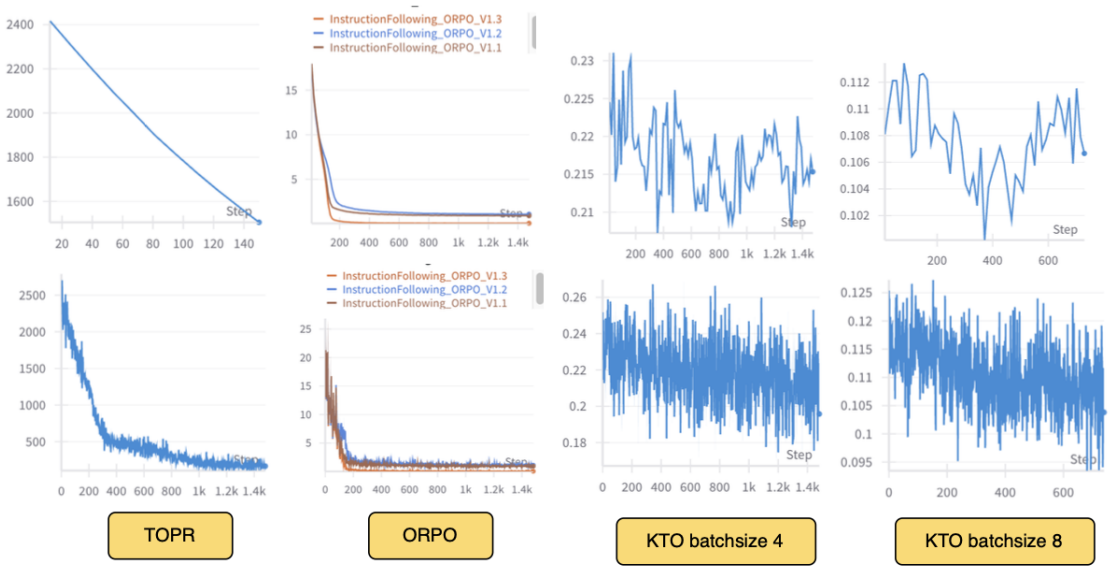

Figure 4: Validation (top) and training (bottom) during one epoch for extension methods TOPR, ORPO and KTO

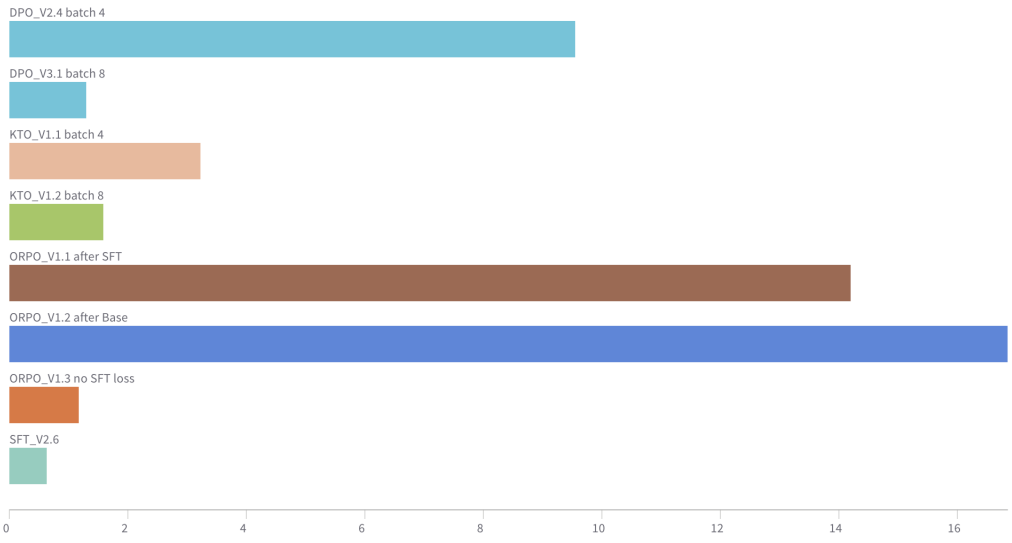

Figure 5: Evaluation loss after training complete. (times 15 due to bug in calculation). RLOO and TOPR validation losses were 230/15 and 21147/15 respectively.

Method	Base Model For Training	Our Model	Win QwenB	Win SFT	Old Board Score	New Board Score	Epoch hh:mm SFT + RL
SFT	Qwen2.5-0.5B	SFT V1.2	54.1	N/A	0.59	N/A	7:20
SFT	Qwen2.5-0.5B	SFT V2.6	83.5	N/A	0.67	N/A	7:40
DPO	SFT V2.6	DPO V2.4	58.6	52.3	N/A	0.1575	7:40 + 2:44
DPO	SFT V2.6	DPO V3.1	63.5	57.4	N/A	0.0625	7:40 + 2:44
RLOO	SFT V2.6	RLOO V1.2	N/A	35.1	N/A	N/A	7:40 + 1:58
TOPR	SFT V2.6	TOPR V1.1	87.8	62.5	0.59	N/A	7:40 + 2:29
ORPO	SFT V2.6	ORPO V1.1	91	87.4	0.725	0.1900	7:40 + 2:08
ORPO	Qwen2.5-0.5	ORPO V1.2	82.5	70.1	N/A	N/A	0:00 + 2:07
ORPO	SFT V2.6	ORPO V1.3	80.5	69.3	N/A	N/A	0:00 + 2:08
KTO	SFT V2.6	KTO V1.1	71.7	62.4	N/A	N/A	7:40 + 2:44

Table 2: Win rate and leaderboard score of our models on Huggingface.

fine-tuned with max length 512 (version V2.4) scored 0.1575 on the leaderboard when also using 512 as the max number of tokens for inference. ORPO fine-tuned on 512 max length (version V1.1) scored 0.19. But when using 1024 max tokens for inference the same DPO model scored only 0.065. Similarly, ORPO's score went down to 0.0675. However, when ORPO was fine-tuned with 1024 max length the score went back up to 0.1825. But finetuning DPO with 1024 brought up the score to only 0.08. Likely doing inference using a longer sequence than it was trained on can lead to incoherent responses that are rated unfavorably by the LLM judge.

5.2 Qualitative Analysis

- ORPO clearly stood out as the most effective and robust method. ORPO V1.1, trained on SFT V2.6, achieved a win rate of 91.0% against the Qwen base and 87.4% against our SFT model, along with the highest leaderboard score (0.725) and new leaderboard score (0.1900). Its success likely stems from its simplicity: by directly modifying the supervised loss with a contrastive odds-ratio term, ORPO simultaneously discourages undesirable outputs and amplifies desirable ones— ithout requiring a reference model. This makes the signal cleaner and less sensitive to hyperparameters.
- More surprisingly, ORPO V1.2 still achieved a 70.1% win rate against our SFT baseline while using only 20% of the total training time. This suggests that ORPO is highly robust to initialization and that its alignment signal is strong enough to guide learning from scratch. ORPO V1.3, which removed the SFT loss entirely and relied solely on the odds-ratio term, still outperformed all DPO, RLOO, and KTO variants—further reinforcing ORPO's stability and standalone effectiveness.
- DPO shows a very unstable validation curve. It only has a slight improvement over SFT when compared to ORPO or TOPR and performed less consistently. One likely reason is DPO's sensitivity to the scale and variance of the log-ratio between model and reference probabilities. This makes the objective unstable when preference signals are noisy. DPO V3.1 showed some improvement as its larger batch size and tuned hyperparameters resulted in a smoother validation loss curve and improved win rate. The model's reduced maximum sequence length of 256, however, negatively impacted its leaderboard performance. This underscores DPO's unpredictable nature and the need for careful hyperparameter and architecture tuning.
- TOPR outperformed DPO in every metric, achieving 87.8% win rate against Qwen and 62.5% vs. SFT, with a leaderboard score of 0.59. While still below ORPO, these results are strong considering no specialized reward model or loss shaping was used and still outperformed DPO on all metrics. This can be attributed to TOPR's unique handling of positive and negative trajectories via truncated importance sampling, which allows it to gracefully unlearn bad behavior while reinforcing good behavior. While it underperformed compared to ORPO, we expect further gains with multi-epoch training, better reward value tuning, and larger batch sizes.

- KTO showed solid performance with a 71.7% win rate against Qwen and 62.4% against SFT. As a lightweight binary preference-based method, KTO is promising for scenarios where fine-grained rewards are unavailable. Its performance suggests strong baseline alignment without requiring pairwise loss terms. KTO's advantage lies in its simplicity and grounding in Prospect Theory: it models loss aversion by weighing penalties for undesirable outputs more heavily. This makes it attractive in settings where pairwise preferences or dense reward labels are unavailable.

- DPO and ORPO models trained on max 512 tokens but inferenced with max 1024 scored poorly on the leaderboard. Another version of DPO, fine-tuned on max 1024, still performed poorly. ORPO fine-tuned on max 1024, however, performed well.

- Despite using the max 512 token fine-tuned SFT model as a base, finetuning ORPO on max 1024 tokens had good results when using max 1024 for inference. However, similarly, DPO performed poorly. This underscores ORPO's advantage of simultaneously performing SFT and preference alignment.

- Our RLOO training run has degraded the model it was trained on. Since it was not required, we did not further dig into why it was the case.

6 Discussion

Preference-based fine-tuning methods such as TOPR, DPO, KTO, and ORPO offer powerful frameworks for aligning reinforcement learning agents with human preferences, yet each comes with notable limitations. TOPR is flexible in handling both positive and negative feedback and avoids reliance on KL regularization. However, it suffers from high variance due to its REINFORCE-style updates and is sensitive to its tapering schedule. This makes tuning critical for stable learning, and we believe performance could improve with further hyperparameter optimization.

DPO, by design, requires strictly ranked preference pairs and cannot accommodate graded feedback, which limits its applicability in nuanced tasks. KTO often struggles with tuning its KL divergence weight: if the weight is too high, the policy may fail to learn and if too low, the policy can diverge. ORPO stands out for its ability to perform off-policy training and to incorporate complex reward models. However, this flexibility comes at the cost of increased complexity. It depends heavily on the quality of reward estimation and incurs additional computational overhead due to off-policy corrections.

Despite these limitations, all of these methods represent important progress in preference-based reinforcement learning, offering pathways toward more interpretable and controllable AI systems. They help bridge the gap between human intent and machine behavior, contributing to data-driven alignment. As our results show, the effectiveness of each method relies on engineering trade-offs between learning stability and scalability. Future research should aim to integrate the strengths of these approaches to ensure reliable and ethical AI behavior at scale.

7 Conclusion

Our findings suggest that ORPO offers the best trade-off among performance, simplicity, and robustness. TOPR and KTO remain promising directions that may yield stronger results with additional tuning, while DPO requires more careful engineering to achieve consistent reliability. Future work could pursue the following directions for improvement:

- TOPR: Extend evaluations to tasks involving richer or more structured preferences, such as multi-turn dialogues or complex reasoning tasks, to assess robustness beyond the UltraFeedback dataset.

- DPO and ORPO: Develop methods that better account for representation learning during training on preference pairs. This challenge is unique to these methods, as KTO and TOPR do not require an equal number of positive and negative responses.

8 Team Contributions

- **Gerardus de Bruijn**: Design and build training loop, dataloaders, implement SFT, DPO, ORPO, KTO, run experiments on gpu, hyper parameter tuning.
- **Nareauphol Liu**: Implement TOPR, Implement RLOO, Hyperparameter fine-tuning, General Code Debugging.
- **Dev Jayram**: Evaluation pipeline for Ultrafeedback, Countdown, Gathered results against SFT, QwenB, and Leaderboard scores

All of us contributed to reviewing extensions and writing poster and final report.

Changes from Proposal The tools extension was dropped and replaced by other fine-tune methods ORPO and KTO. Countdown was implemented for SFT, DPO and RLOO but dropped due to technical challenges especially with RLOO.

References

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. KTO: Model Alignment as Prospect Theoretic Optimization. arXiv:2402.01306 [cs.LG] https://arxiv.org/abs/2402.01306

Jiwoo Hong, Noah Lee, and James Thorne. 2024. ORPO: Monolithic Preference Optimization without Reference Model. arXiv:2403.07691 [cs.CL] https://arxiv.org/abs/2403.07691

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. arXiv:2305.18290 [cs.LG] https://arxiv.org/abs/2305.18290

Nicolas Le Roux, Marc G. Bellemare, Jonathan Lebensold, Arnaud Bergeron, Joshua Greaves, Alex Fréchette, Carolyne Pelletier, Eric Thibodeau-Laufer, Sándor Toth, and Sam Work. 2025. Tapered Off-Policy REINFORCE: Stable and efficient reinforcement learning for LLMs. arXiv:2503.14286 [cs.LG] https://arxiv.org/abs/2503.14286

A Additional Experiments

A.1 Training details

Two more training experiments were done to evaluate the effect of training with a larger batch size and with a larger maximum sequence length of 1024, using a more powerful gpu.

A.1.1 Hardware details

One instance gpu_1x_gh200 on cloud.lambda.ai, NVIDIA GH200 Superchips (ARM64 + H100) with 64 VCPU and 96GB memory.

A.1.2 Hyper parameter search

Optuna was used for searching optimal learning rate and regularizers for DPO and ORPO.

- DPO: 25 trials for learning rate in $(1e^{-6}, 5e^{-5})$ and beta in (0.1, 0.5).
- ORPO: 20 trials for learning rate in $(1e^{-6}, 5e^{-5})$ and lambda in (0.03, 0.3).

A.1.3 Training details

Both DPO and ORPO were trained for 4 epochs with batch size 12 and maximum sequence length 1024.

- DPO: learning rate $4e^{-6}$, $\beta = 0.1$.

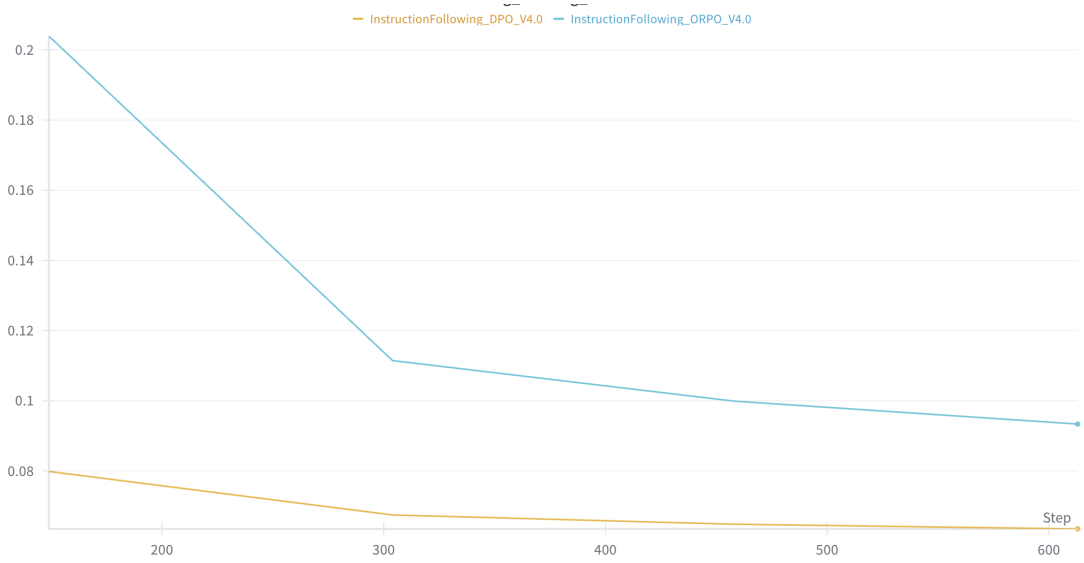

Figure 6: Training loss for DPO and ORPO over 4 epochs with batchsize 12, max 1024 tokens.

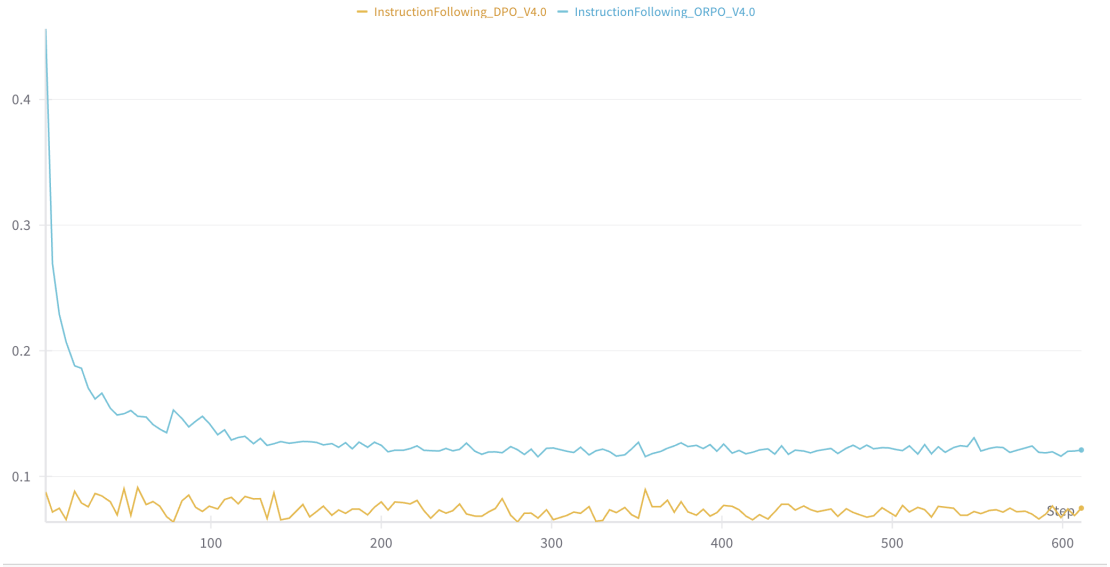

Figure 7: Validation loss for DPO and ORPO over 4 epochs with batchsize 12, max 1024 tokens.

- ORPO: learning rate $5e^{-5}$ and $\lambda = 0.06$.

Since the Optuna optimization found 0.1 as the best value for DPO and 0.1 was also the lower range in the search, likely an even lower value for beta could have been optimal. Similarly, a higher learning rate than $5e^{-5}$ could have been found for ORPO.

A.1.4 Training Results

Figure 6 shows the training loss for DPO and ORPO over 4 epochs with batchsize 12, max 1024 tokens.

Figure shows the validation loss for DPO and ORPO over 4 epochs with batchsize 12, max 1024 tokens

Method	Base Model For Training	Our Model	SFT max tokens	Finetune max tokens	Inference max tokens	Name	New Board Score
DPO	SFT V2.6	DPO V2.4	512	512	1024	Haha DPO	0.065
DPO	SFT V2.6	DPO V2.4	512	512	512	Haha DPO 512	0.1575
DPO	SFT V2.6	DPO V3.1	512	256	1024	Haha DPO 3.1	0.0625
DPO	SFT V2.6	DPO V4.0	512	1024	1024	Haha DPO 1024	0.08
ORPO	SFT V2.6	ORPO V1.1	512	512	512	Haha ORPO 512	0.1900
ORPO	SFT V2.6	ORPO V1.1	512	512	1024	Haha ORPO V1.1	0.0675
ORPO	SFT V2.6	ORPO V4.0	512	1024	1024	Haha ORPO 1024	0.1825

Table 3: New leaderboard scores for models with various max tokens for SFT training, finetuning and inference.

Method	Base Model For Training	Our Model	HuggingFace Model
SFT	Qwen2.5-0.5B	SFT V1.2	guydebruyn/InstructionFollowing_SFT_V1.2
SFT	Qwen2.5-0.5B	SFT V2.6	guydebruyn/InstructionFollowing_SFT_V2.6
DPO	SFT V2.6	DPO V2.4	guydebruyn/InstructionFollowing_DPO_V2.4
DPO	SFT V2.6	DPO V3.1	guydebruyn/InstructionFollowing_DPO_V3.1
DPO	SFT V2.6	DPO V4.0	guydebruyn/InstructionFollowing_DPO_V4.0
RLOO	SFT V2.6	RLOO V1.2	guydebruyn/InstructionFollowing_RLOO_V1.2
TOPR	SFT V2.6	TOPR V1.1	guydebruyn/InstructionFollowing_TOPR_V1.1
ORPO	SFT V2.6	ORPO V1.1	guydebruyn/InstructionFollowing_ORPO_V1.1
ORPO	Qwen2.5-0.5	ORPO V1.2	guydebruyn/InstructionFollowing_ORPO_V1.2
ORPO	SFT V2.6	ORPO V1.3	guydebruyn/InstructionFollowing_ORPO_V1.3
ORPO	SFT V2.6	ORPO V4.0	guydebruyn/InstructionFollowing_ORPO_V4.0

Table 4: List of checkpoints pushed to Huggingface for each model version.

A.2 Extended Leaderboard results

Table 3 lists the results of all uploads to the leaderboard for each version, and the corresponding max tokens that have been used for SFT, finetuning and inference to get the scores.

A.3 Model checkpoints

Table 4 lists the Huggingface checkpoints of the models we have trained.

B Implementation Details

- A single Python script trainer.py that executes all trainings, including hyper parameter tuning with Optuna. All script parameters are detailed in README.md file.
- By default all datasets and models are loaded from Huggingface, including the base models.
- In addition to storing locally on disk, a model can be pushed to Huggingface for reuse.
- If indicated, standard output and metrics are logged in Weights&Biases, in addition to logging to local directory.
- A separate evaluator.py script performs all evaluations. It is used to calculate the winrate of the model against a reference model as well as generate json lines output for the leaderboard submission.
- VLLM server instances as spun up for the model and reference model it is evaluated against
- Inference calls to get the reward for a response are made as API calls to to nvidia/llama-3.1-nemotron-70b-reward hosted on openai.