STOP TRAINING ON PAIRS YOU'VE ALREADY LEARNED: POLICY-AWARE PAIR SELECTION FOR PREFERENCE OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning from human feedback (RLHF) is the standard approach to aligning language models with human preferences, and in practice it is most often performed offline with direct methods such as DPO that learn from preference pairs alone. What such methods learn depends heavily on which pairs they see. Following the Delta Learning Hypothesis, which pairs responses from a large and a small model from the same model class, a strong recent rule (MaxMin) selects the best and worst responses from a candidate pool for each prompt. We show that DPO with MaxMin data selection saturates the training signal quickly. In particular, the policy quickly separates the best from the worst responses, its implicit reward margins grow, and the objective's sigmoid flattens, resulting in a small gradient norm and weight updates. We propose ActiveDelta, an adaptive selection rule that follows the policy being trained: for each prompt, it picks the largest-quality-gap pair according to an external oracle feedback which the current policy has not yet confidently separated, keeping gaps large while avoiding saturation. ActiveDelta consistently outperforms tuned MaxMin selection on downstream benchmarks and reward-model scores on a held-out set of prompts. Crucially, ActiveDelta keeps improving where MaxMin plateaus; on the benchmark suite it matches MaxMin with one-sixth of the prompts. A practical variant that ranks candidates with a learned reward model instead of an oracle and queries the reward oracle only for the selected pair retains these gains, which makes ActiveDelta applicable when scoring every response is too costly.

1 INTRODUCTION

Reinforcement learning from human feedback (RLHF) is the standard methodology for aligning large language models (LLMs) with human preferences: a reward model is trained on preference data, and a policy is then optimized against that reward model with proximal policy optimization (PPO) (Ouyang et al., 2022; Schulman et al., 2017). Direct preference optimization (DPO) removes the separate reward model and the RL stage and fits the policy directly to the preference pairs (Rafailov et al., 2024). This makes DPO simpler and more stable, but it also makes the outcome depend almost entirely on which pairs the policy is trained on.

Modern preference data generation pipelines therefore sample many responses per prompt from a pool of models, score them with a reward model that we refer to as the *oracle*, and keep a single pair per prompt (Tunstall et al., 2023; Cui et al., 2024; Lambert et al., 2025; Walsh et al., 2025; Swiss AI Initiative, 2026). Following the Delta Learning Hypothesis (Geng et al., 2025), which attributes the gains of preference tuning to the quality difference between paired responses, a strong recent rule pairs each prompt's best and worst responses (MaxMin). DPO training on MaxMin pairs, however, saturates after a small fraction of the training data. The cause lies in the DPO objective, in which the gradient of each pair is scaled by a weight that decays toward zero as the policy learns to distinguish the chosen response from the rejected one (Section 3).

Given several sampled responses per prompt and their oracle rewards, we study which pair the optimizer should train on at each step. Two quantities describe a pair. The *oracle reward gap* is computed once by the annotator and measures how distinguishable the pair is in principle. The

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

implicit reward margin changes during training and measures how distinguishable the pair is to the current policy. ActiveDelta selects, for each prompt, the pair with the largest oracle gap whose implicit margin under the current policy is still below a threshold, and recomputes the selection as the policy changes. The policy therefore keeps receiving the strongest annotator signal it has not yet absorbed. We evaluate the rule for DPO. It is defined for any pairwise loss whose per-pair gradient weight decays as the margin grows, such as SimPO (Meng et al., 2024), the preference term of ORPO (Hong et al., 2024) and the hinge loss of SLiC-HF (Zhao et al., 2023), with both the margin and the threshold expressed in the units of the respective loss (Appendix D).

Querying the oracle for every candidate response is costly when the oracle is a large reward model, an LLM judge or a human annotator. Active preference data generation pipelines such as ActiveUltraFeedback (Melikidze et al., 2026) address this cost with active learning. They train a reward model with uncertainty estimates on the preferences obtained from the oracle so far and use it to choose, for each prompt, the pair of responses that the oracle annotates. ActiveDelta extends to this setting by ranking the candidate pairs with the reward gap estimated by this learned reward model in place of the oracle gap.

Our contributions are as follows.

- We demonstrate that DPO training on MaxMin pairs saturates early, which we attribute to the DPO gradient weight vanishing on exactly the pairs MaxMin selects (Section 3).
- We propose ActiveDelta, a pair-level selection rule that keeps this weight bounded away from zero. Against a MaxMin baseline tuned separately at every prompt budget, it improves downstream benchmark performance and reward-model scores under the annotating oracle, and continues to improve past the point where MaxMin plateaus (Section 6.1).
- We show that the gain holds at every prompt budget from 2.5k to 60k with both methods tuned per budget, and that ActiveDelta matches the 60k MaxMin model with one-sixth of the prompts (Section 6.2).
- With rewards estimated by the learned reward model of ActiveUltraFeedback, ActiveDelta retains its gain and outperforms the pair selection rule of that pipeline (Section 6.3).

2 Related Work

Pair construction. UltraFeedback (Cui et al., 2024) established the practice of sampling many responses per prompt from a pool of models and scoring each with a reward model, which leaves the choice of the training pair to an acquisition function. The delta learning hypothesis (Geng et al., 2025) attributes the gains of preference tuning to the quality gap between the paired responses, and Lee et al. (2026) find that filtering pairs by this gap improves data efficiency. Xiao et al. (2025) observe that pairing the highest- and lowest-scored responses degrades as the number of on-policy samples per prompt grows, and place the rejected response two standard deviations below the mean reward. Curry-DPO (Pattnaik et al., 2024) trains on several pairs per prompt, ordered from large to small oracle gap over successive passes. These rules, like the acquisition functions compared by ActiveUltraFeedback (Melikidze et al., 2026), fix the pairs before training and do not consult the policy being trained.

Data selection. Other methods select a subset of the available pairs or prompts before training. Selective DPO (Gao et al., 2025) removes examples that exceed the model's capacity; Qi et al. (2025) keep the pairs with the smallest implicit reward gap under a separately aligned model; Guo et al. (2025) keep the prompts with the largest preference variance, which bounds the norm of the DPO gradient; and Zhang et al. (2026) score pairs by their influence on validation performance and find that the value of a pair depends on the model being trained. AMPO (Gupta et al., 2025) and MASS-DPO (Surana et al., 2026) select, within each prompt, the responses of a multi-response loss, by coverage of reward extremes and semantic clusters and by Fisher information at the initial policy, respectively. SeRA (Ko et al., 2024) and Deng et al. (2025) retain the pairs with a large implicit margin, or a large combination of external and implicit margins, and use the margin as a proxy for label reliability. Two methods select during training: SamS (Huang et al., 2025) schedules the samples of each batch with a contextual bandit that observes the model's learning feedback, and SAGE (Wu et al., 2026a) re-scores fixed pairs as the policy evolves and excludes those of low

gradient utility from backpropagation. In reinforcement learning with verifiable rewards, prompts whose sampled responses all receive the same reward have zero advantage and are filtered during training (Yu et al., 2025), and the prompts that carry the strongest learning signal shift as training proceeds (Wu et al., 2026b). ActiveDelta also selects during training, among the candidate pairs of every prompt and by a closed-form rule: the largest oracle gap subject to an implicit margin below C. It obtains label reliability from the oracle gap and uses the margin only to exclude pairs whose gradient weight has vanished, which are the high-margin pairs that SeRA and Deng et al. retain.

Active preference learning. Active preference learning selects which responses to annotate under a labeling budget. APL (Muldrew et al., 2024) combines the predictive entropy of the policy with the absolute implicit margin, prioritizing confidently ranked pairs on the hypothesis that they are the most likely to be confidently wrong, and ActiveDPO (Lin et al., 2026) replaces this certainty term with a gradient-based uncertainty. In iterative DPO, Yang et al. (2024) annotate the pairs with the smallest implicit reward margin and find them more useful than pairs with large margins or random pairs. Other criteria include the regret under a linear reward model (Das et al., 2024), a D-optimal design on the linearized DPO objective (Kveton et al., 2025), a design criterion derived from bounds on the suboptimality of the DPO-trained policy (Han et al., 2026) and the Sharpe ratio of the DPO gradient (Belakaria et al., 2025); in online DPO with current models, uncertainty-based selection yields negligible gains over random selection (Oh et al., 2026). These methods select before the label is known and therefore cannot use the sign of the implicit margin. Applied within a prompt at training time, APL's certainty term is the APL-style baseline of Section 6.1. The variant of Section 6.3 recovers the labeling setting by querying the oracle only for the selected pair.

Loss reweighting and online training. Other methods keep the pairs and change their weight in the loss. β-DPO (Wu et al., 2024) adapts β per batch, WPO (Zhou et al., 2024) reweights off-policy pairs by their likelihood under the current policy, Uni-DPO (Peng et al., 2026) reweights pairs by their quality and the model's current performance on them, and Gate-DPO (Mouiche & Bahi, 2026) attenuates the gradient of rejected responses that have reached low probability, which delays the saturation of the loss. These methods are complementary to ActiveDelta, which keeps the loss fixed and changes the pair. Online DPO regenerates the responses from the current policy (Guo et al., 2024; Dong et al., 2024; Xu et al., 2024); on-policy sampling improves preference fine-tuning empirically (Tajwar et al., 2024) and theoretically (Kim et al., 2026), at the cost of generation during training. Section 6.4 shows that online MaxMin pairs do not saturate but end below offline ActiveDelta.

3 Background

RLHF and DPO. A language model defines a policy $\pi_\theta(y \mid x)$ over responses y to a prompt x. RLHF assumes that preferences between two responses follow a latent reward function $r(x, y)$ through the Bradley–Terry model (Bradley & Terry, 1952), $p(y_w \succ y_l \mid x) = \sigma\big(r(x, y_w) - r(x, y_l)\big)$, where σ is the logistic function. It fits a reward model to preference pairs and then trains the policy with reinforcement learning to maximize the learned reward, with a KL penalty of weight β towards a reference policy π_{ref}, usually the supervised fine-tuned (SFT) model (Ziegler et al., 2019; Ouyang et al., 2022). DPO (Rafailov et al., 2024) optimizes the same objective without a reward model or reinforcement learning, by minimizing on each preference pair the loss

$$\mathcal{L}_{\text{DPO}} = -\log \sigma(m_\theta), \qquad m_\theta = \beta\left[\log \tfrac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \tfrac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right], \tag{1}$$

where y_w is the preferred and y_l the rejected response, and m_θ, the difference between the *implicit rewards* $\beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ of the two responses, is the *implicit reward margin* of the pair.

Gradient weight. The gradient of the DPO loss is $\nabla_\theta \mathcal{L}_{\text{DPO}} = -\,w(m_\theta)\,\nabla_\theta m_\theta$, where $w(m_\theta) = \sigma(-m_\theta)$ is the *gradient weight* of the pair. It equals 0.5 at initialization, where $m_\theta = 0$, and vanishes as the margin grows. We call a loss with this property *saturating*.

Preference data. For each prompt x in a prompt set $\mathcal{D}$, K candidate responses Y_x are sampled from a pool of models and scored by an *oracle* $s(x, y)$, which can be a human annotator or a reward

model. A pair is oriented such that $s(x, y_w) > s(x, y_l)$ and has the *oracle reward gap* $\Delta_s = s(x, y_w) - s(x, y_l)$. MaxMin selects, for every prompt, the pair with the largest Δ_s.

Saturation. Downstream scores of MaxMin runs plateau after roughly a quarter of training (Section 6.1). We attribute this plateau to *gradient saturation*. A MaxMin pair has the largest oracle reward gap of its prompt, and both of its responses are generated by models other than the policy. The policy separates such pairs within a few updates, and since MaxMin selects one for every prompt, the batch-mean gradient weight $\bar{w}$, and with it the effective number of pairs per update, falls early in training (Figure 1). ActiveDelta (Section 4) keeps the gradient weight bounded away from zero by selection alone, without changing the loss.

4 ACTIVEDELTA

In this section, we introduce ActiveDelta, a novel preference learning algorithm which keeps MaxMin's preference for large oracle gaps but skips pairs the current policy has already separated providing no training signal. We develop it for DPO, and Appendix D extends the idea to other direct preference optimization algorithms. For a prompt x with candidate responses Y_x, orient every pair such that $s(x, y_a) > s(x, y_b)$, where s is the oracle or, in Section 6.3, a learned proxy for it; each pair then has a reward gap $\Delta_s(y_a, y_b) = s(x, y_a) - s(x, y_b)$ and the implicit reward margin of Equation 1,

$$m_\theta(y_a, y_b) = \beta \left[\log \frac{\pi_\theta(y_a|x)}{\pi_{\text{ref}}(y_a|x)} - \log \frac{\pi_\theta(y_b|x)}{\pi_{\text{ref}}(y_b|x)} \right].$$

ActiveDelta selects, for every prompt, the pair with the largest reward gap among those whose implicit reward margin is below a threshold C:

$$(y_w, y_l) = \arg \max_{(y_a, y_b) \in Y_x^2} \Delta_s(y_a, y_b) \quad \text{subject to} \quad m_\theta(y_a, y_b) < C. \tag{2}$$

The threshold C is placed on m_θ, the argument of the sigmoid, and is the rule's only hyperparameter; every selected pair has gradient weight above $\sigma(-C)$. Setting $C = \infty$ removes the constraint and recovers MaxMin. If no pair satisfies the constraint, the rule falls back to the pair of highest- and lowest-scored responses.

Algorithm 1 gives the training loop. Each step draws a batch of prompts from $\mathcal{D}$ and processes every prompt in it independently. The reward model first scores the K candidate responses of the prompt (line 4). The implicit margin of every pair is then computed under the current policy (line 5); since m_θ is a difference of per-response log-ratios, this requires evaluating the policy once per candidate response, after which the $K(K-1)/2$ pair margins are differences of those K values. Selection (line 6) implements Equation 2: the pairs are traversed in decreasing order of reward gap and the first one whose margin is below C is taken. If every pair is above the threshold, the highest- and lowest-scored responses are paired (line 8). The selected pairs of the batch form the training batch for one DPO gradient step (line 12). Since margins are computed under the current policy, the pairs that satisfy the constraint change over training: at initialization every margin is zero and the rule selects MaxMin pairs; as the policy pushes the margins of large-reward-gap pairs beyond C, the rule selects pairs with smaller reward gaps. The reward model s is either pretrained and fixed, or the learned reward model of an active preference data generation pipeline, updated iteratively alongside the execution of ActiveDelta (Sections 5 and 6.3).

5 EXPERIMENTAL SETUP

To evaluate ActiveDelta as described in Section 4, we generate our own preference data: a prompt set, a pool of candidate responses per prompt, and an oracle score for every response. The prompt set comes from the Apertus 1.5 preference dataset (Swiss AI Initiative, 2026), which reuses the prompts of Dolci-Instruct-DPO (Olmo et al., 2026): about 246k prompts covering coding, mathematics, science, constrained instruction following, safety, general chat and multilingual requests. We use only the prompts; for each one we generate $K = 28$ candidate responses from a pool of 28 open-weight models spanning several model families and scales from 0.5B to 675B parameters (Appendix E), and score every response with Skywork-Reward-V2-Llama-3.1-8B (Liu et al., 2026b), which serves

Algorithm 1 ActiveDelta

Require: policy π_θ, reference policy π_{ref}, β, reward model s, prompt set $\mathcal{D}$ with candidate responses Y_x, threshold C

for each batch $\mathcal{B}$ of prompts from $\mathcal{D}$ **do**
  $\mathcal{S} \leftarrow \emptyset$
  for each prompt $x \in \mathcal{B}$ **do**
    Obtain $s(x, y)$ for all $y \in Y_x$
    Compute $m_\theta(y_a, y_b)$ for all pairs in Y_x
    $(y_w, y_l) \leftarrow \arg\max\{\Delta_s(y_a, y_b) : m_\theta(y_a, y_b) < C\}$
    if no such pair exists **then**
      $(y_w, y_l) \leftarrow (\arg\max_y s(x, y),\ \arg\min_y s(x, y))$
    end if
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(x, y_w, y_l)\}$
  end for
  Update θ with one gradient step on $-\log \sigma(m_\theta)$ over $\mathcal{S}$
end for
return π_θ

as the oracle throughout. To keep the experiments tractable, each prompt budget is a random subset of this pool, 60k prompts for the main comparisons.

All experiments align Olmo-3-7B-Instruct-SFT (Olmo et al., 2026), or Apertus-1.5-8B-SFT in Section 6.5, with full-parameter training for a single epoch over the prompts. The main baseline is DPO on MaxMin pairs from the same candidate responses. At every prompt budget, both methods are tuned separately over grids of equal size covering the learning rate, β and, for ActiveDelta, the threshold C (Appendix A); "tuned MaxMin" refers to the best configuration of this search. Two further baselines change only the selection rule: a random candidate pair, and the APL-style pair of largest absolute implicit margin (Section 2), both oriented by the oracle and tuned over the same learning rates and β values.

For the experiments of Section 6.3 we use the active preference data generation pipeline of ActiveUltraFeedback (Melikidze et al., 2026) directly, with the same prompts, candidate responses and oracle as above. The pipeline's learned reward model is a Bayesian linear head on the frozen last-hidden-state features of Skywork-Reward-V2-Qwen3-4B (Liu et al., 2026b), updated with oracle scores as they are collected. It takes the place of the reward model s in Algorithm 1, and the oracle scores only the selected pair, two calls per prompt instead of K. The baseline is the pipeline's own acquisition function, DeltaUCB (Melikidze et al., 2026), which selects the pair with the largest optimistic reward gap under the head, at the same two oracle calls per prompt. The head, its update schedule and the pipeline configuration are described in Appendix F.

For the experiments of Section 6.4 we use the online DPO implementation of Apertus 1.5 (Swiss AI Initiative, 2026). For each prompt, 8 responses are sampled from the current policy and scored by the same oracle, and the highest- and lowest-scored responses form the training pair, as in the Apertus 1.5 setup. Configuration details are in Appendix G.

6 Results

We evaluate models in two ways. First, on seven post-training benchmarks: GSM8K (CoT) (Cobbe et al., 2021) for mathematical reasoning, HumanEval (Chen et al., 2021) for code generation, BBH (Suzgun et al., 2023) for multi-step reasoning, IFEval (Zhou et al., 2023) for verifiable instruction following, AlpacaEval 2 (Dubois et al., 2025) for open-ended helpfulness, reported as length-controlled win rate, TruthfulQA (MC2) (Lin et al., 2022) for truthfulness, and BBQ (Parrish et al., 2022) for social bias. Their unweighted mean is the aggregate we use for hyperparameter selection and as the primary metric. Second, on alignment to the oracle: we generate responses to the 1,865 prompts of RewardBench 2 (Malik et al., 2025), a held-out set disjoint from the training prompts, score them with the reward model that annotated the training pool, and report the mean reward and the win rate against the SFT model. Since the policy is trained toward this reward model, these scores measure how well each selection rule achieves the objective it optimizes. Section 6.1 com-

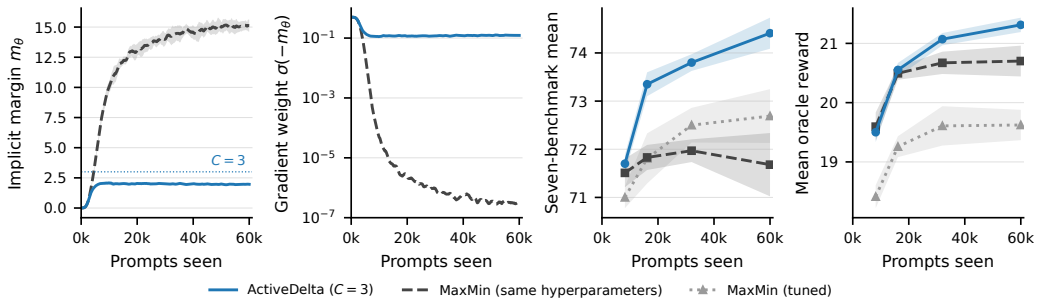

Figure 1: **MaxMin saturates; ActiveDelta holds the gradient weight.** One epoch over 60k prompts; every batch consists of prompts the policy has not seen. From left: batch-mean implicit margin m_θ of the selected pairs; gradient weight $\sigma(-m_\theta)$ at this margin (log scale); seven-benchmark mean and mean oracle reward on RewardBench 2 prompts at four checkpoints. ActiveDelta and MaxMin with the same hyperparameters both use lr 2×10^{-6}, $\beta = 0.05$ and effective batch 32; tuned MaxMin, shown in the two right panels, uses $\beta = 0.1$ with the same learning rate and batch size. Bands are 95% confidence intervals over five training seeds.

pares ActiveDelta with MaxMin on 60k prompts, Section 6.2 across prompt budgets, Section 6.3 inside an active preference data generation pipeline, Section 6.4 under online DPO, and Section 6.5 on a second base model.

6.1 ActiveDelta against MaxMin

Figure 1 shows both methods over training on the 60k prompts. MaxMin's batch-mean implicit margin reaches 10 within the first 10k prompts, and the gradient weight at this margin falls below 10^{-3} for the rest of the epoch. ActiveDelta's constraint binds at the same point and holds this weight at 0.12. With the same hyperparameters as ActiveDelta, MaxMin's seven-benchmark mean stops improving after 16k prompts, while ActiveDelta keeps improving. Tuned MaxMin ($\beta = 0.1$) keeps improving slowly but stays below ActiveDelta at every checkpoint. At the end of the epoch, with each method at its best configuration, ActiveDelta ($C = 4$) is 1.9 points above MaxMin on the seven-benchmark mean (Table 1). The gain comes from AlpacaEval, BBQ and TruthfulQA; GSM8K, HumanEval and BBH stay on par, and IFEval is about two points lower. Oracle alignment shows the same picture. MaxMin plateaus after about a quarter of the epoch, and ActiveDelta ends 0.8 higher in mean reward and 2.3 points higher in win rate at the same hyperparameters, and by a wider margin against tuned MaxMin (Table 2, Appendix C). Across thresholds, downstream performance peaks around $C = 4$, while oracle reward keeps rising with C; beyond $C = 4$, the threshold therefore trades downstream performance for alignment to the oracle (Appendices A, C and I).

Selection baselines. The random and APL-style baselines (Section 5) both score about a point below tuned MaxMin and three below ActiveDelta (Table 3, Appendix A). The APL-style rule keeps the oracle gap large but selects pairs the policy has already separated, whose gradient weight is near zero; random selection keeps the weight, but its pairs have less than half the oracle gap. Neither property suffices on its own.

6.2 Sample Efficiency across Prompt Budgets

Table 4 repeats the comparison at 10k, 20k and 60k prompts, with both methods tuned at each budget, and Table 8 adds 2.5k and 5k. ActiveDelta is ahead at every budget, and the gap grows from half a point at 2.5k and 5k to 1.5–2 points from 10k on; at 10k, a paired comparison on identical prompts and at the same hyperparameters confirms the gain (Appendix A). Between 5k and 20k prompts, MaxMin improves little while ActiveDelta continues to improve, consistent with saturation. ActiveDelta at 10k prompts matches MaxMin at 60k, with one sixth of the data. On

Table 1: **Downstream benchmarks at 60k prompts.** Best configuration per method (MaxMin at $\beta = 0.1$, ActiveDelta at $\beta = 0.05$; both lr 2×10^{-6}, effective batch 32), with the SFT model for reference. Online MaxMin (Section 6.4) trains on the highest- against the lowest-scored of 8 responses sampled from the current policy, at the best configuration of its own grid ($\beta = 0.001$, lr 2×10^{-6}, 64 prompts per step; single seed). Means over five training seeds; $\pm$ is the standard deviation of the seven-benchmark mean across seeds. Bold marks the best value in each row.

		MaxMin		ActiveDelta	
Benchmark	**SFT**	Online	Offline	$C = 3$	$C = 4$
GSM8K (CoT)	75.3	79.8	85.5	**86.5**	85.9
HumanEval	91.8	94.5	94.2	94.2	**94.7**
BBH	68.1	70.2	68.8	**70.9**	69.3
IFEval	81.3	79.1	**83.5**	82.3	81.6
AlpacaEval 2 (LC)	34.7	**61.6**	52.4	57.1	59.2
TruthfulQA (MC2)	50.7	50.8	54.4	56.8	**57.0**
BBQ	61.9	62.1	70.2	73.5	**74.4**
Average	66.2	71.2	72.7 ± 0.5	74.5 ± 0.4	**74.6** ± 0.2

Table 2: **Oracle alignment at 60k prompts.** Mean oracle reward and win rate against the SFT model on the 1,865 RewardBench 2 prompts at the final checkpoint; mean $\pm$ SD over five training seeds. All rows use lr 2×10^{-6} and effective batch 32; ActiveDelta and MaxMin with the same hyperparameters use $\beta = 0.05$, and tuned MaxMin, the baseline of Table 1, uses $\beta = 0.1$.

Method	**Mean reward**	**Win rate (%)**
MaxMin (same hyperparameters)	20.70 ± 0.21	74.6 ± 0.7
MaxMin (tuned)	19.62 ± 0.21	71.3 ± 1.2
ActiveDelta ($C = 3$)	21.31 ± 0.10	75.8 ± 1.1
ActiveDelta ($C = 4$)	21.53 ± 0.14	76.9 ± 0.5
ActiveDelta ($C = 5$)	21.74 ± 0.20	77.3 ± 1.0
ActiveDelta ($C = 6$)	21.63 ± 0.22	76.9 ± 0.8
ActiveDelta ($C = 7$)	**21.78** ± 0.13	**77.8** ± 0.8

oracle alignment, ActiveDelta is also ahead at each budget, and its lead grows from 0.9 to 1.9 in mean reward and from 1.8 to 5.6 points in win rate (Appendix C).

6.3 ActiveDelta in an Active Preference Data Generation Pipeline

The results so far assume that the oracle scores all K candidate responses of every prompt. In an active preference data generation pipeline, a learned reward model ranks the candidates instead, and the oracle scores only the selected pair (Section 5). ActiveDelta keeps its gain in this setting (Table 5, Appendix F). On the same prompts and at the same hyperparameters, it is 2.4 points above the pipeline's own acquisition function, DeltaUCB. With a fourteenth of the oracle calls, it is only slightly below oracle-ranked ActiveDelta.

6.4 Online DPO

Section 6.1 shows saturation on fixed offline pairs. We test whether it persists when the pairs are the policy's own samples, by running online DPO with rollouts from the current policy, scored by the same oracle and paired by MaxMin (Section 5). Saturation does not occur in this setting: the implicit margin of the selected pairs stays small for the whole epoch, the gradient weight remains close to its initial value of 0.5, and the seven-benchmark mean rises at every checkpoint (Appendix G). Online DPO nevertheless ends below offline training. Its best configuration, tuned over its own grid, reaches 71.2, against 72.7 for tuned MaxMin and 74.6 for ActiveDelta (Table 1), and its gain comes mostly

Table 3: **Selection baselines at 60k prompts.** Seven-benchmark mean with each rule at its best configuration over the same grid (lr $\in \{1, 2, 4\} \times 10^{-6}$, $\beta \in \{0.05, 0.1\}$, effective batch 32), and the mean oracle gap Δ_s and mean implicit margin m_θ of the selected pairs over the epoch. Baseline rows are single seed (42); MaxMin and ActiveDelta are means over five seeds.

Selection rule	Best config	Mean of 7	Δ_s	m_θ
Random	4×10^{-6}, 0.05	71.7	10.8	2.1
Largest $\lvert m_\theta \rvert$ (APL-style)	4×10^{-6}, 0.05	71.8	24.0	9.6
MaxMin	2×10^{-6}, 0.1	72.7 ± 0.5	36.6	16.9
ActiveDelta ($C = 4$)	2×10^{-6}, 0.05	**74.6** ± 0.2	22.0	2.6

Table 4: **ActiveDelta against tuned MaxMin across prompt budgets.** Best configuration per method and budget; budgets are random prompt subsets, each tuned separately (Table 8, which also covers 2.5k and 5k prompts). Oracle alignment as in Table 2, with ActiveDelta at $C = 4$. Mean $\pm$ SD over five training seeds.

	Seven-benchmark mean		Oracle mean reward		Win rate vs. SFT (%)	
Prompts	MaxMin	ActiveDelta	MaxMin	ActiveDelta	MaxMin	ActiveDelta
10k	71.7 ± 0.3	**73.2** ± 0.4	20.08 ± 0.20	**20.98** ± 0.09	73.1 ± 0.5	**74.9** ± 0.6
20k	71.8 ± 0.1	**73.7** ± 0.2	19.81 ± 0.18	**20.89** ± 0.22	72.6 ± 1.1	**74.9** ± 0.9
60k	72.7 ± 0.5	**74.6** ± 0.2	19.62 ± 0.21	**21.53** ± 0.14	71.3 ± 1.2	**76.9** ± 0.5

from AlpacaEval, while TruthfulQA and BBQ change little and IFEval falls below the SFT model. On-policy pairs avoid saturation but come from the policy alone; ActiveDelta avoids it on the offline pool, without generating responses during training, and ends higher.

6.5 Generalization to a Different Base Model

We repeat the 60k comparison with Apertus-1.5-8B-SFT (Swiss AI Initiative, 2026) in place of Olmo-3-7B-Instruct-SFT, with the same prompts, candidate responses and oracle, and both methods tuned separately (Appendix H). With both methods at MaxMin's best configuration, ActiveDelta leads by 0.9 points, against 1.9 on Olmo (Table 6). The gain comes from BBQ and TruthfulQA, as on Olmo, but ActiveDelta's AlpacaEval score, above MaxMin's on Olmo, is below it on Apertus.

7 Limitations

Data quality. ActiveDelta depends on the quality of the candidate responses and of the oracle that scores them. The responses must be diverse enough that pairs with a large oracle gap remain once the policy has separated the extreme ones. The oracle must be strong enough to order the gaps reliably: among the pairs below the threshold, ActiveDelta takes the one with the largest oracle gap. We test a single pool of 28 generators and a single oracle.

Compute. Selection evaluates the current policy on all K candidate responses of each prompt, K forward passes without a backward pass; over a single epoch, this is one forward pass of the policy over the candidate pool. On 60k prompts and the same hardware, an ActiveDelta epoch takes 7.6 hours against 3.1 hours for MaxMin; the difference is the evaluation of the candidates (Appendix A). Our implementation evaluates the frozen reference on every candidate at every step as well; its log-probabilities can instead be computed once for the pool and shared across runs. We compare the methods at equal prompts and gradient steps, not at equal compute. More MaxMin training does not close the gap: a second epoch over the same prompts leaves its score nearly unchanged at the best learning rate and lowers it at a higher one (Table 11), since its pairs are already saturated.

Table 5: **ActiveDelta with a learned reward model.** Seven-benchmark aggregate at 60k prompts; all rows use lr 2×10^{-6}, $\beta = 0.05$ and effective batch 32; mean $\pm$ SD over five training seeds. Rows ranked by the learned reward model share prompts and seeds; the oracle-ranked rows are the runs of Section 6.1, on a different random 60k draw from the same pool.

Ranking	Selection	Oracle calls / prompt	Aggregate
Oracle	MaxMin	28	71.7 ± 0.5
Oracle	ActiveDelta ($C = 4$)	28	**74.6** ± 0.2
Learned reward model	DeltaUCB	2	71.9 ± 0.2
Learned reward model	ActiveDelta ($C = 4$)	2	**74.3** ± 0.3

Table 6: **Two base models at 60k prompts.** Same prompts, candidate responses and oracle for both; ActiveDelta at $C = 4$. Means over five training seeds; $\pm$ is the SD of the average. Olmo columns are those of Table 1; on Apertus, both methods use MaxMin's best configuration, lr 6×10^{-6}, $\beta = 0.05$ and effective batch 32.

	Olmo-3-7B-Instruct-SFT		Apertus-1.5-8B-SFT	
Benchmark	**MaxMin**	**ActiveDelta**	**MaxMin**	**ActiveDelta**
GSM8K (CoT)	85.5	85.9	80.7	80.5
HumanEval	94.2	94.7	88.1	87.5
BBH	68.8	69.3	66.9	67.3
IFEval	83.5	81.6	82.8	81.7
AlpacaEval 2 (LC)	52.4	59.2	63.0	60.1
TruthfulQA (MC2)	54.4	57.0	53.6	57.1
BBQ	70.2	74.4	66.1	73.1
Average	72.7 ± 0.5	**74.6** ± 0.2	71.6 ± 0.3	**72.5** ± 0.3

Threshold. C must be tuned jointly with the learning rate and β, but the loss narrows the search. The weight of a pair at the threshold is $\sigma(-C)$, which falls from 0.05 at $C = 3$ to below 0.001 at $C = 7$; larger thresholds admit pairs that are nearly saturated (Figure 3). The best downstream threshold lay between 3 and 6 in every setting we ran; we have no rule that selects it within this range.

Scope. We evaluate the rule with DPO only, although the argument covers any saturating loss (Appendix D). The main results use one base model and a single epoch; on the second base model of Section 6.5, the gain is smaller. Oracle alignment is an in-distribution metric, measured with the reward model that labeled the training data; the downstream benchmarks guard against over-optimizing it.

8 Conclusion

DPO training on MaxMin pairs saturates early. The policy learns to distinguish these pairs within a fraction of an epoch, after which their gradient weight vanishes. ActiveDelta keeps the large oracle gap that makes MaxMin effective and uses the implicit reward margin to skip the pairs the policy has already learned. With this one change, it improves downstream benchmarks and alignment to the oracle over a tuned MaxMin baseline and matches it with a sixth of the prompts. It retains this gain when a learned reward model replaces the oracle for ranking the candidates.

AI use statement

In this work, we used generative AI tools to implement methods, to interpret results and to provide feedback on the experiments, and to generate synthetic data sets: the candidate responses of Section 5 are produced by language models as part of the method. We have not used generative AI

tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, design research methodology or experiments, or clean and reformat datasets, and proofs, translation and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create and edit software code, create scientific figures, summarize and identify relevant literature, suggest the structure of the paper, draft parts of the paper, edit it for readability and format references. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

Section 5 describes the data, the models and the training protocol, and Section 6 the evaluation; Equation 2 and Algorithm 1 specify ActiveDelta. Olmo-3-7B-Instruct-SFT, the oracle and the 28 generators are publicly available (Appendix E). Appendix A gives the hyperparameter grids, fixed training settings and seeds, Appendix J the evaluation settings, and Appendices F and G the configurations of the active pipeline and of online DPO. The code for data generation, training and analysis, and the annotated candidate pool, will be made public after the review process.

Ethics statement

This work uses publicly released prompts, models and reward models and involves no human subjects or personal data. ActiveDelta trains a policy toward an automated oracle and inherits the oracle's biases and errors; because it optimizes the oracle more effectively than MaxMin (Section 6.1), it can also exploit those errors more effectively. We therefore select hyperparameters on downstream benchmarks that the oracle does not score, and report BBQ for social bias and the Safety domain of RewardBench 2; ActiveDelta scores higher than MaxMin on both. Like any preference optimization method, ActiveDelta could be used to train a model toward harmful preferences, given a reward model that encodes them. The annotated candidate pool that we will release contains model-generated responses to safety-related prompts, some of which may be harmful; it will be released under the licenses of its source prompts and models.

References

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, et al. SmolLM2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Arcee AI. Trinity-Mini. https://huggingface.co/arcee-ai/Trinity-Mini, December 2025a.

Arcee AI. Trinity-Nano-Preview. https://huggingface.co/arcee-ai/Trinity-Nano-Preview, December 2025b.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A general theoretical paradigm to understand learning from human preferences, 2023. URL https://arxiv.org/abs/2310.12036.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, et al. SmolLM3: Smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, July 2025.

Syrine Belakaria, Joshua Kazdan, Charles Marx, Chris Cundy, Willie Neiswanger, Sanmi Koyejo, Barbara E. Engelhardt, and Stefano Ermon. Sharpe ratio-guided active learning for preference optimization in rlhf, 2025. URL https://arxiv.org/abs/2503.22137.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. ULTRAFEEDBACK: Boosting language models with scaled AI feedback. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 9722–9744. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/cui24f.html.

Nirjhar Das, Souradip Chakraborty, Aldo Pacchiano, and Sayak Ray Chowdhury. Active preference optimization for sample efficient RLHF, 2024. URL https://arxiv.org/abs/2402.10500.

Xun Deng, Han Zhong, Rui Ai, Fuli Feng, Zheng Wang, and Xiangnan He. Less is more: Improving LLM alignment via preference data selection. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2502.14560.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf, 2024. URL https://arxiv.org/abs/2405.07863.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators, 2025. URL https://arxiv.org/abs/2404.04475.

Chengqian Gao, Haonan Li, Liu Liu, Zeke Xie, Peilin Zhao, and Zhiqiang Xu. Principled data selection for alignment: The hidden risks of difficult examples, 2025. URL https://arxiv.org/abs/2502.09650.

Leo Gao, Jonathan Tow, Baber Abbasi, et al. The Language Model Evaluation Harness, July 2024. URL https://zenodo.org/records/12608602.

Gemma Team. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Scott Geng, Hamish Ivison, Chun-Liang Li, Maarten Sap, Jerry Li, Ranjay Krishna, and Pang Wei Koh. The delta learning hypothesis: Preference tuning on weak data can yield strong gains. In *Second Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2507.06187.

Aaron Grattafiori, Abhimanyu Dubey, et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Jiacheng Guo, Zihao Li, Jiahao Qiu, Yue Wu, and Mengdi Wang. On the role of preference variance in preference optimization, 2025. URL https://arxiv.org/abs/2510.13022.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, Johan Ferret, and Mathieu Blondel. Direct language model alignment from online ai feedback, 2024. URL https://arxiv.org/abs/2402.04792.

Taneesh Gupta, Rahul Madhavan, Xuchao Zhang, Chetan Bansal, and Saravan Rajmohan. Ampo: Active multi-preference optimization for self-play preference selection, 2025. URL https://arxiv.org/abs/2502.18293.

Jiangze Han, Vineet Goyal, and Will Ma. Which pairs to compare for LLM post-training?, 2026. URL https://arxiv.org/abs/2606.19607.

Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model, 2024. URL https://arxiv.org/abs/2403.07691.

Zixuan Huang, Yikun Ban, Lean Fu, Xiaojie Li, Zhongxiang Dai, Jianxin Li, and Deqing Wang. Adaptive batch-wise sample scheduling for direct preference optimization. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2506.17252.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, et al. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Juno Kim, Jihun Yun, Jason D. Lee, and Kwang-Sung Jun. Coverage improvement and fast convergence of on-policy preference learning, 2026. URL https://arxiv.org/abs/2601.08421.

Jongwoo Ko, Saket Dingliwal, Bhavana Ganesh, Sailik Sengupta, Sravan Bodapati, and Aram Galstyan. Sera: Self-reviewing and alignment of large language models using implicit reward margins, 2024. URL https://arxiv.org/abs/2410.09362.

Branislav Kveton, Xintong Li, Julian McAuley, Ryan Rossi, Jingbo Shang, Junda Wu, and Tong Yu. Active learning for direct preference optimization, 2025. URL https://arxiv.org/abs/2503.01076.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

Chia-Hsuan Lee, Mingyang Zhou, Renkun Ni, Zelei Cheng, Sihui Dai, Supriyo Chakraborty, Shixiong Zhang, Sambit Sahu, and William Campbell. Decomposing the delta: What do models actually learn from preference pairs?, 2026. URL https://arxiv.org/abs/2604.08723.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

Xiaoqiang Lin, Arun Verma, Zhongxiang Dai, Daniela Rus, See-Kiong Ng, and Bryan Kian Hsiang Low. ActiveDPO: Active direct preference optimization for sample-efficient alignment. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2505.19241.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, et al. Ministral 3, 2026a. URL https://arxiv.org/abs/2601.08584.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. Skywork-reward-v2: Scaling preference data curation via human-ai synergy, 2026b. URL https://arxiv.org/abs/2507.01352.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. RewardBench 2: Advancing reward model evaluation, 2025. URL https://arxiv.org/abs/2506.01937.

Pedro Henrique Martins, Patrick Fernandes, Jo ao Alves, et al. EuroLLM: Multilingual language models for Europe, 2024. URL https://arxiv.org/abs/2409.16235.

Davit Melikidze, Marian Schneider, Jessica Lam, Martin Wertich, Ido Hakimi, Barna Pásztor, and Andreas Krause. ActiveUltraFeedback: Efficient preference data generation using active learning, 2026. URL https://arxiv.org/abs/2603.09692.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward, 2024. URL https://arxiv.org/abs/2405.14734.

Mistral AI. Introducing Mistral 3. https://mistral.ai/news/mistral-3, December 2025a.

Mistral AI. Mistral Small 3. https://mistral.ai/news/mistral-small-3, January 2025b.

Inoussa Mouiche and Abderaouf Bahi. Gradient-gated DPO: Stabilizing preference optimization in language models, 2026. URL https://arxiv.org/abs/2605.02626.

William Muldrew, Peter Hayes, Mingtian Zhang, and David Barber. Active preference learning for large language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 36577–36590. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/muldrew24a.html.

Giyeong Oh, Junghyun Lee, Jaehyun Park, Youngjae Yu, Wonho Bae, and Junhyug Noh. Random is hard to beat: Active selection in online DPO with modern LLMs, 2026. URL https://arxiv.org/abs/2604.02766.

Team Olmo, :, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2026. URL https://arxiv.org/abs/2512.13961.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Alicia Parrish, Angelica Chen, Nikita Nangia, Vishakh Padmakumar, Jason Phang, Jana Thompson, Phu Mon Htut, and Samuel R. Bowman. BBQ: A hand-built bias benchmark for question answering. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2086–2105, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.165. URL https://aclanthology.org/2022.findings-acl.165/.

Pulkit Pattnaik, Rishabh Maheshwary, Kelechi Ogueji, Vikas Yadav, and Sathwik Tejaswi Madhusudhan. Curry-dpo: Enhancing alignment using curriculum learning & ranked preferences, 2024. URL https://arxiv.org/abs/2403.07230.

Shangpin Peng, Weinong Wang, Zhuotao Tian, Senqiao Yang, Xing Wu, Haotian Xu, Chengquan Zhang, Takashi Isobe, Baotian Hu, and Min Zhang. Uni-DPO: A unified paradigm for dynamic preference optimization of LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2506.10054.

Xuan Qi, Rongwu Xu, and Zhijing Jin. Difficulty-based preference data selection by DPO implicit reward gap, 2025. URL https://arxiv.org/abs/2508.04149.

Qwen Team. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, February 2026.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024. URL https://arxiv.org/abs/2305.18290.

Miguel Moura Ramos, Duarte M. Alves, Hippolyte Gisserot-Boukhlef, et al. EuroLLM-22B: Technical report, 2026. URL https://arxiv.org/abs/2602.05879.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Rohan Surana, Xintong Li, Sheldon Yu, Yiran Jenny Shen, Chuhan Wang, Tong Yu, Prithviraj Ammanabrolu, Jingbo Shang, Julian McAuley, and Junda Wu. MASS-DPO: Multi-negative active sample selection for direct policy optimization, 2026. URL https://arxiv.org/abs/2605.10784.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Swiss AI Initiative. Apertus-v1.5-8B. https://huggingface.co/swiss-ai/Apertus-v1.5-8B, 2026.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of llms should leverage suboptimal, on-policy data, 2024. URL https://arxiv.org/abs/2404.14367.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. Zephyr: Direct distillation of LM alignment, 2023. URL https://arxiv.org/abs/2310.16944.

Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James Validad Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 furious (COLM's version). In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=2ezugTT9kU.

Hui Wu, Hengyi Cai, Jinman Zhao, Xinran Chen, Ziheng Li, Zhejun Zhao, Shuaiqiang Wang, Yuchen Li, and Dawei Yin. Not all preferences deserve gradients: Understanding gradient utility in offline reasoning alignment. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026a. URL https://arxiv.org/abs/2602.01207.

Jiahao Wu, Ning Lu, Shengcai Liu, Kun Wang, Yanting Yang, Baijiong Lin, Chen Jason Zhang, Li Qing, and Ke Tang. Train at moving edge: Online-verified prompt selection for efficient RL training of large reasoning model, 2026b. URL https://arxiv.org/abs/2603.25184.

Junkang Wu, Yuexiang Xie, Zhengyi Yang, Jiancan Wu, Jinyang Gao, Bolin Ding, Xiang Wang, and Xiangnan He. β-dpo: Direct preference optimization with dynamic β, 2024. URL https://arxiv.org/abs/2407.08639.

Yao Xiao, Hai Ye, Linyao Chen, Hwee Tou Ng, Lidong Bing, Xiaoli Li, and Roy Ka wei Lee. Finding the sweet spot: Preference data construction for scaling preference optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025. URL https://arxiv.org/abs/2502.16825.

Jin Xu, Zhifang Guo, Hangrui Hu, et al. Qwen3-omni technical report, 2025. URL https://arxiv.org/abs/2509.17765.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. Is dpo superior to ppo for llm alignment? a comprehensive study, 2024. URL https://arxiv.org/abs/2404.10719.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Sen Yang, Leyang Cui, Deng Cai, Xinting Huang, Shuming Shi, and Wai Lam. Not all preference pairs are created equal: A recipe for annotation-efficient iterative preference learning, 2024. URL https://arxiv.org/abs/2406.17312.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, et al. DAPO: An open-source LLM reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Zizhuo Zhang, Qizhou Wang, Shanshan Ye, Jianing Zhu, Jiangchao Yao, Bo Han, and Masashi Sugiyama. Towards understanding valuable preference data for large language model alignment. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.13212.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J. Liu. Slic-hf: Sequence likelihood calibration with human feedback, 2023. URL https://arxiv.org/abs/2305.10425.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

Wenxuan Zhou, Ravi Agrawal, Shujian Zhang, Sathish Reddy Indurthi, Sanqiang Zhao, Kaiqiang Song, Silei Xu, and Chenguang Zhu. Wpo: Enhancing rlhf with weighted preference optimization, 2024. URL https://arxiv.org/abs/2406.11827.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Training Settings and Hyperparameter Grids

Training settings. All offline runs use the AdamW optimizer (Loshchilov & Hutter, 2019) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and no weight decay, a linear learning-rate schedule with warmup over the first 10% of steps, bfloat16 precision, and a maximum length of 4096 tokens for prompt and response together. Both methods run in the same training loop, which implements Algorithm 1 with $C = \infty$ for MaxMin; it adapts the training loop of ActiveUltraFeedback (Melikidze et al., 2026) and uses components of TRL.[1] All runs use nodes with four NVIDIA GH200 GPUs each and finish within 12 hours. On 60k prompts and two nodes, a MaxMin epoch takes 3.1 hours, or 25 GPU-hours, and an ActiveDelta epoch 7.6 hours, or 61 GPU-hours, of which 4.8 hours evaluate the policy and the reference on the candidate responses. Multi-seed results use seeds 42–46, and single-seed runs use seed 42.

Fallback. The fallback of Algorithm 1 (line 8) applies only if every pair of consecutively ranked responses has a margin of at least C. Since margins are differences of per-response log-ratios, the highest- and lowest-scored responses are then separated by at least $(K-1)C$, which is 81 at $C = 3$; MaxMin, which trains on exactly these pairs, ends the epoch at a batch-mean margin of 15 (Table 18).

Grids. Table 7 lists every run of the search at 60k prompts. Both searches have 38 runs: MaxMin varies the learning rate, β and the effective batch size, and ActiveDelta the learning rate, β and the threshold C at effective batch 32. Table 8 gives the grids below 60k prompts and the configuration selected at every budget.

Paired comparison at 10k prompts. Both methods were trained with five seeds on the same 10k prompts at ActiveDelta's selected configuration (Table 8); paired by seed, ActiveDelta leads by 2.15 ± 0.30 (SE; paired t-test, $p = 0.002$).

Table 7: **Hyperparameter grids on the 60k pool.** Seven-benchmark mean, one seed per cell. ActiveDelta was run at effective batch 32 (1875 steps), and MaxMin at effective batch 32, 64 and 128 (1875, 938 and 469 steps); dashes mark settings that were not run. Bold marks the ActiveDelta cells above MaxMin at batch 32 with the same learning rate and β; the best threshold exceeds MaxMin in 8 of the 9 settings run for both methods. Two further MaxMin runs at $\beta = 0.2$ and batch 32 reach 71.71 (lr 2×10^{-6}) and 72.83 (lr 4×10^{-6}), and two further ActiveDelta runs at $\beta = 0.05$ and lr 1×10^{-6} reach 72.83 at $C = 2$ and 72.79 at $C = 7$. MaxMin ranges over 5.49 points across its grid (67.73–73.22), almost three times the gap between the tuned methods at 60k prompts. The one unstable region is $\beta = 0.025$ with lr 2×10^{-6}, where the looser thresholds degrade sharply.

		ActiveDelta, batch 32				**MaxMin, batch**		
β	**Learning rate**	$C = 3$	$C = 4$	$C = 5$	$C = 6$	32	64	128
0.025	5×10^{-7}	**72.72**	**72.51**	**72.06**	**71.90**	71.81	71.19	68.41
	1×10^{-6}	**73.11**	**72.69**	**72.39**	**71.56**	71.08	71.95	70.57
	2×10^{-6}	**71.66**	66.80	62.88	62.54	70.67	71.52	71.75
	4×10^{-6}	–	–	–	–	70.27	71.24	71.31
0.05	5×10^{-7}	**71.93**	**72.01**	**72.23**	**72.33**	71.56	70.66	68.14
	1×10^{-6}	**73.51**	**73.64**	**73.68**	**73.57**	72.60	71.59	70.10
	2×10^{-6}	**74.34**	**74.21**	**73.65**	**73.56**	71.39	72.15	71.28
	4×10^{-6}	–	–	–	–	72.09	71.30	72.06
0.1	5×10^{-7}	70.67	70.31	70.84	70.90	70.91	69.77	67.73
	1×10^{-6}	71.18	**72.12**	71.58	**72.48**	71.71	70.91	69.09
	2×10^{-6}	71.90	72.49	72.56	**73.71**	73.22	71.70	70.21
	4×10^{-6}	–	–	–	–	72.61	72.54	71.33

[1] https://github.com/huggingface/trl

Table 8: **Selected configuration and seven-benchmark mean per method and prompt budget**, the runs of Table 4 and the 2.5k and 5k budgets. The selected ActiveDelta configurations all use $\beta = 0.05$. Below 60k prompts, each method is tuned over nine runs at effective batch 16, one seed each: MaxMin over $\beta \in \{0.025, 0.05, 0.1\}$ and lr $\in \{1, 2, 4\} \times 10^{-6}$, ActiveDelta over $C \in \{3, 4, 5\}$ and the same learning rates. The 60k grids are those of Table 7. Mean $\pm$ SD over five training seeds for the selected configurations; ActiveDelta at 2.5k and 5k is a single run.

		MaxMin			**ActiveDelta**		
Budget	**Batch**	Learning rate	β	Mean of 7	Learning rate	C	Mean of 7
2.5k	16	4×10^{-6}	0.1	71.0 ± 0.2	4×10^{-6}	4	**71.5**
5k	16	2×10^{-6}	0.05	71.6 ± 0.3	4×10^{-6}	3	**72.1**
10k	16	2×10^{-6}	0.05	71.7 ± 0.3	4×10^{-6}	4	**73.2** ± 0.4
20k	16	1×10^{-6}	0.05	71.8 ± 0.1	2×10^{-6}	3	**73.7** ± 0.2
60k	32	2×10^{-6}	0.1	72.7 ± 0.5	2×10^{-6}	4	**74.6** ± 0.2

A.1 Selection baselines

Tables 9 and 10 give every run behind Table 3: both baseline rules over the same learning rates and β values, at effective batch 32 on the 60k pool, one seed (42) per cell. At each of the six settings, both rules score below MaxMin with the same hyperparameters (Table 7). The diagnostics in Table 10 hold across the grid: the APL-style rule selects pairs with oracle gaps between 21.7 and 24.0 at margins between 6.8 and 11.9, and the random rule selects pairs with a gap of 10.8 at margins between 1.1 and 2.5. Under the APL-style rule, the share of selections the policy ranks the wrong way falls as the learning rate rises, from 4.8% to 2.5%: the criterion finds fewer confidently wrong pairs the faster the policy learns, which is the degradation Lin et al. (2026) describe.

Table 9: **Selection baselines, per-benchmark scores** (60k pool, effective batch 32, seed 42). HEval: HumanEval; AE-LC: AlpacaEval 2 (LC); TQA: TruthfulQA (MC2). Bold marks the best run of each rule.

Rule	lr, β	GSM8K	HEval	BBH	IFEval	AE-LC	TQA	BBQ	Mean of 7
Random	1×10^{-6}, 0.05	77.4	93.5	69.7	81.3	46.2	52.8	64.5	69.34
	1×10^{-6}, 0.1	76.4	94.0	69.6	82.3	43.5	52.2	63.4	68.77
	2×10^{-6}, 0.05	79.8	93.4	70.8	83.4	48.3	54.6	66.7	71.00
	2×10^{-6}, 0.1	78.9	93.2	69.3	82.8	45.8	52.6	64.0	69.51
	4×10^{-6}, 0.05	83.5	92.8	70.5	84.7	46.8	55.1	68.6	**71.71**
	4×10^{-6}, 0.1	80.9	93.0	69.1	83.0	41.9	52.9	64.3	69.30
Largest $\lvert m_\theta \rvert$	1×10^{-6}, 0.05	76.8	94.2	71.7	81.7	49.7	52.1	65.8	70.29
	1×10^{-6}, 0.1	77.1	93.7	68.7	82.4	47.7	51.7	64.1	69.34
	2×10^{-6}, 0.05	77.6	94.3	71.0	80.8	51.6	53.8	67.7	70.99
	2×10^{-6}, 0.1	78.3	93.1	70.3	81.5	49.3	52.7	65.7	70.13
	4×10^{-6}, 0.05	80.8	94.1	67.8	81.9	52.8	55.7	69.3	**71.76**
	4×10^{-6}, 0.1	80.0	92.7	70.0	80.6	43.6	54.1	65.8	69.54

A.2 Two-epoch training and intermediate checkpoints

Table 11 reports MaxMin trained for a second epoch over the same prompts. Table 12 gives the seven-benchmark means of Figure 1 at its four checkpoints, with the paired difference between ActiveDelta and MaxMin with the same hyperparameters.

Table 10: **Selection baselines, diagnostics of the selected pairs**, averaged over the epoch: mean oracle gap Δ_s, mean implicit margin m_θ, the share of selections the policy ranks the wrong way, and the mean AlpacaEval response length in words. The random rule's gap is the mean over unordered candidate pairs and does not depend on the policy.

Rule	lr, β	Δ_s	m_θ	$m_\theta < 0$	Words
Random	1×10^{-6}, 0.05	10.8	1.05	–	308
	1×10^{-6}, 0.1	10.8	1.28	–	277
	2×10^{-6}, 0.05	10.8	1.58	–	302
	2×10^{-6}, 0.1	10.8	1.85	–	265
	4×10^{-6}, 0.05	10.8	2.07	–	309
	4×10^{-6}, 0.1	10.8	2.52	–	273
Largest $\lvert m_\theta \rvert$	1×10^{-6}, 0.05	21.7	6.77	4.8%	317
	1×10^{-6}, 0.1	22.1	8.76	4.5%	303
	2×10^{-6}, 0.05	23.4	8.47	3.2%	345
	2×10^{-6}, 0.1	23.6	10.44	3.1%	319
	4×10^{-6}, 0.05	24.0	9.63	2.5%	379
	4×10^{-6}, 0.1	23.8	11.94	2.8%	323

Table 11: **Second epoch of MaxMin.** MaxMin trained for two epochs over the same 60k prompts ($\beta = 0.1$, effective batch 32, one linear decay spanning both passes), against one epoch at the same setting and against ActiveDelta at one epoch. Single seed (42) for the two-epoch rows; the one-epoch five-seed mean is given for reference. AE-LC: AlpacaEval 2 (LC).

Method	Epochs, lr	GSM8K	BBH	IFEval	AE-LC	Mean of 7
MaxMin	2, 1×10^{-6}	82.5	68.8	81.9	54.9	71.9
MaxMin	2, 2×10^{-6}	86.6	68.2	84.8	52.6	73.3
MaxMin	2, 4×10^{-6}	70.7	63.8	82.4	48.7	70.3
MaxMin	1, 2×10^{-6} (seed 42)	86.5	69.0	81.3	54.2	73.0
MaxMin	1, 2×10^{-6} (5 seeds)	85.5	68.8	83.5	52.4	72.7 ± 0.5
ActiveDelta ($C = 3$)	1, 2×10^{-6} (5 seeds)	86.5	70.9	82.3	57.1	74.5 ± 0.4

Table 12: **Seven-benchmark mean at intermediate checkpoints**, 60k prompts, effective batch 32, lr 2×10^{-6}; mean $\pm$ SD over five seeds. ActiveDelta and MaxMin with the same hyperparameters use $\beta = 0.05$, and tuned MaxMin uses $\beta = 0.1$. The paired difference, ActiveDelta minus MaxMin with the same hyperparameters, is computed within seed on the same prompts.

	8k	16k	32k	60k
ActiveDelta ($C = 3$)	71.7 ± 0.2	73.4 ± 0.2	73.8 ± 0.1	74.4 ± 0.3
MaxMin (same hyperparameters)	71.5 ± 0.2	71.8 ± 0.2	72.0 ± 0.2	71.7 ± 0.5
MaxMin (tuned)	71.0 ± 0.2	71.8 ± 0.4	72.5 ± 0.3	72.7 ± 0.5
Paired difference	+0.2	+1.5	+1.8	+2.7

B PER-BENCHMARK SCORES

Table 13 gives the per-benchmark scores of the best cell of each method at each β in the grids of Appendix A. BBQ and TruthfulQA are the only benchmarks on which ActiveDelta is higher at every β, by 4 to 5 and about 2 points respectively. Within a setting, the threshold trades BBH for AlpacaEval: at $\beta = 0.05$ and lr 2×10^{-6}, raising C from 3 to 6 lowers BBH from 70.5 to 64.0 and raises AlpacaEval from 57.3 to 60.8, while the average falls from 74.34 to 73.56. The ranking of thresholds therefore depends on the weighting of the benchmarks.

Table 13: **Per-benchmark scores for the best cell at each** β, effective batch 32, 60k pool, seed 42. Within each β block, the two columns are the best MaxMin and the best ActiveDelta cell over the learning rates and, for ActiveDelta, the thresholds. ActiveDelta was run only at batch 32; allowing MaxMin all three batch sizes changes only the $\beta = 0.025$ entry, from 71.81 to 71.95 (batch 64). Bold marks the higher average in each β block.

	$\beta = 0.025$		$\beta = 0.05$		$\beta = 0.1$	
Benchmark	MaxMin	ActiveDelta	MaxMin	ActiveDelta	MaxMin	ActiveDelta
GSM8K (CoT)	83.7	81.9	84.5	85.9	86.7	83.2
HumanEval	94.9	94.3	94.6	94.9	95.4	94.8
BBH	64.4	63.5	65.4	70.5	69.3	70.8
IFEval	81.3	84.3	83.4	83.0	82.4	83.0
AlpacaEval 2 (LC)	58.9	61.2	58.2	57.3	56.7	55.2
TruthfulQA (MC2)	53.6	55.8	54.3	56.7	54.3	56.4
BBQ	65.9	70.8	67.9	72.1	67.7	72.6
Average	71.81	**73.11**	72.60	**74.34**	73.22	**73.71**

C ORACLE ALIGNMENT

Figure 2 shows oracle alignment over training for every threshold at 10k, 20k and 60k prompts. At 10k and 20k prompts, the thresholds that are best downstream ($C = 3, 4$) match MaxMin with the same hyperparameters and $C \geq 5$ exceeds it; at 60k, every threshold exceeds it. Tuned MaxMin is below MaxMin with the same hyperparameters on both metrics at every budget. Table 14 gives the final checkpoint at 60k prompts on each of the six RewardBench 2 domains.

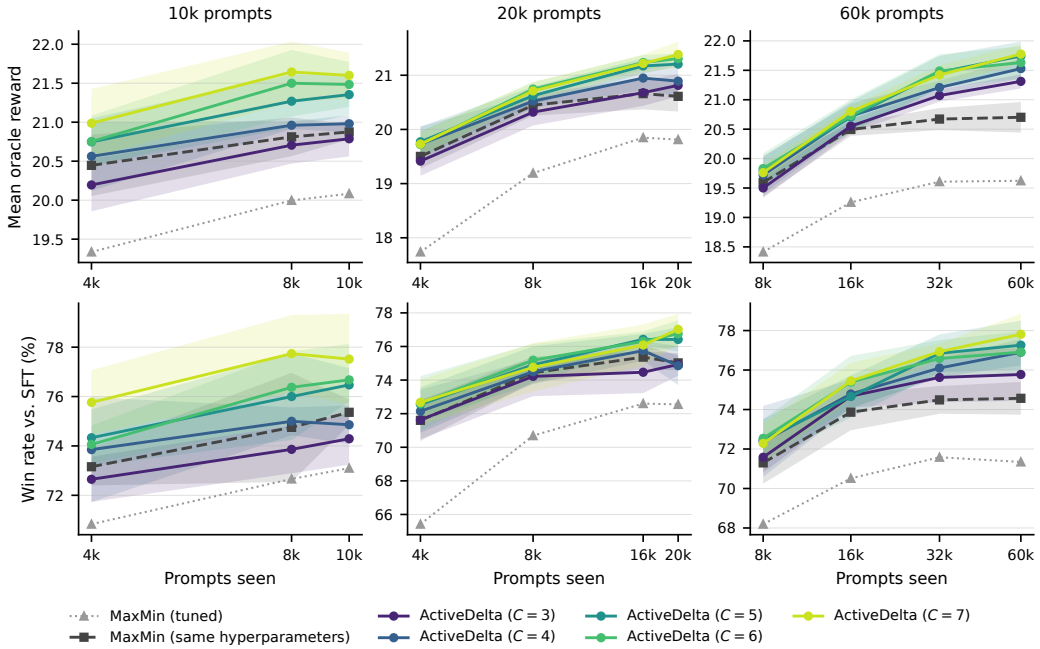

Figure 2: Oracle alignment for every threshold at 10k, 20k and 60k prompts, against MaxMin with the same hyperparameters (dashed) and tuned MaxMin (dotted). Mean oracle reward (top) and win rate against the SFT model (bottom) on RewardBench 2 prompts. Means over five training seeds, four at the intermediate checkpoints of both MaxMin configurations at 10k and of tuned MaxMin at 20k; bands are 95% confidence intervals. Budgets differ in learning rate and effective batch size; curves are therefore comparable within a panel but not across panels.

Table 14: **Per-domain oracle scores at the final checkpoint** (60k pool). Mean reward and win rate against SFT on the six RewardBench 2 domains; five-seed means for tuned MaxMin, $C = 3$ and $C = 4$, one seed for the other rows. The $C = 3$ and $C = 4$ rows are the runs of Table 1, on a different 60k draw than the corresponding rows of Table 2; weighted by domain size, they give a mean reward of 21.39 and 21.60. Safety and Precise IF carry the largest gains; Math changes little.

	Focus	Factuality	Safety	Math	Precise IF	Ties
Mean reward						
MaxMin (same hyperparameters)	26.45	17.28	21.76	19.35	13.26	14.06
MaxMin (tuned)	25.63	16.75	19.84	18.79	12.46	15.59
ActiveDelta ($C = 3$)	26.86	17.72	24.31	19.35	13.56	14.97
ActiveDelta ($C = 4$)	26.90	17.91	24.93	19.25	13.72	14.91
ActiveDelta ($C = 5$)	27.03	18.53	25.04	18.92	13.40	14.12
ActiveDelta ($C = 6$)	27.02	18.65	25.26	18.79	13.72	14.01
Win rate vs. SFT (%)						
MaxMin (same hyperparameters)	75.9	74.0	83.9	65.9	71.9	44.6
MaxMin (tuned)	72.3	69.7	80.7	63.9	69.6	49.2
ActiveDelta ($C = 3$)	76.1	73.2	89.7	67.2	76.4	46.7
ActiveDelta ($C = 4$)	76.2	74.8	89.8	66.7	76.1	44.0
ActiveDelta ($C = 5$)	75.3	75.9	88.0	65.0	74.1	38.2
ActiveDelta ($C = 6$)	76.0	77.5	90.4	63.4	76.3	41.2

D Gradient Weight of DPO and Other Pairwise Losses

DPO. Figure 3 plots the gradient weight $\sigma(-m_\theta)$ against the implicit margin. The weight is 0.27 at $m_\theta = 1$, 0.12 at $m_\theta = 2$, 0.05 at $m_\theta = 3$, 0.018 at $m_\theta = 4$ and 0.007 at $m_\theta = 5$; a pair at margin 5 contributes about 1.3% of the gradient it contributed at initialization.

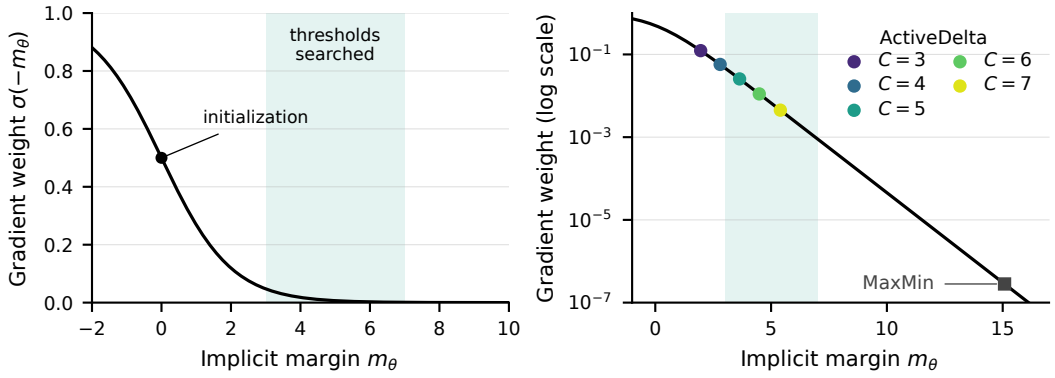

Figure 3: **DPO gradient weight against the implicit margin.** Left: every pair starts at $m_\theta = 0$ with weight 0.5, and the weight is below 0.05 from $m_\theta = 3$ on. Right, on a log scale: the batch-mean margin at the end of the epoch for MaxMin and for ActiveDelta at each threshold (60k prompts, five seeds; Table 18). The shaded band is the range of thresholds we search, $3 \le C \le 7$, over which the weight at the threshold falls from 0.05 to 0.001.

Other pairwise losses. SimPO (Meng et al., 2024) and the preference term of ORPO (Hong et al., 2024) apply the same loss $-\log \sigma(m)$ to a different margin: the scaled difference of length-normalized log-likelihoods minus a target margin for SimPO, and the log odds ratio of the two responses for ORPO. Their gradient weight is therefore $\sigma(-m)$, as for DPO, and the rule applies with C placed on their margin. The calibration loss of SLiC-HF (Zhao et al., 2023) is a hinge, $\max(0, \delta - m)$, on the difference of log-likelihoods; its weight is 1 below δ and 0 above, and with

$C = \delta$ the rule selects only pairs with nonzero gradient. IPO (Azar et al., 2023) is not saturating: its loss $(m - 1/(2\tau))^2$ has a weight linear in m that is negative above the target $1/(2\tau)$, and a pair with a large margin is pushed back toward the target. The failure mode of MaxMin therefore does not arise under IPO in the same form, and a cap on the margin has no direct analogue there.

E Model Pool

Table 15 lists the generators of the candidate responses and their licenses. They belong to the Qwen2.5 (Qwen Team, 2024), Qwen3 (Yang et al., 2025), Qwen3-Omni (Xu et al., 2025), Qwen3.5 (Qwen Team, 2026), EuroLLM (Martins et al., 2024; Ramos et al., 2026), Mistral 3 (Mistral AI, 2025a; Liu et al., 2026a), Mistral Small 3 (Mistral AI, 2025b), Mixtral (Jiang et al., 2024), Gemma 3 (Gemma Team, 2025), Llama 3.2 (Grattafiori et al., 2024), SmolLM2 (Allal et al., 2025), SmolLM3 (Bakouch et al., 2025) and Trinity (Arcee AI, 2025a;b) families. Each generator produces one response per prompt at temperature 1.0, with the thinking mode of hybrid reasoning models disabled and prompt and response together limited to 4096 tokens. A response is marked as truncated if prompt and response together exceed 3900 tokens; the margin below 4096 accounts for the tokenizer of the policy, which differs from those of the generators. Prompts with fewer than 20 non-truncated responses are discarded, truncated responses are excluded from selection, and identical responses are counted once; on the 60k prompts, a prompt retains about 26 of the 28 responses on average.

Table 15: **Generators of the candidate responses**, ordered by parameter count. Active parameters are given for the mixture-of-experts models. Both Trinity models were released under Apache 2.0 and relicensed under OpenMDW-1.1 in May 2026. Gemma: Gemma Terms of Use; Llama 3.2: Llama 3.2 Community License; Qwen Research: Qwen Research License.

Model	Parameters (B)	Active (B)	License
Qwen2.5-0.5B-Instruct	0.5	–	Apache 2.0
Qwen3-0.6B	0.6	–	Apache 2.0
gemma-3-1b-it	1	–	Gemma
Llama-3.2-1B-Instruct	1	–	Llama 3.2
Qwen2.5-1.5B-Instruct	1.5	–	Apache 2.0
Qwen3-1.7B	1.7	–	Apache 2.0
EuroLLM-1.7B-Instruct	1.7	–	Apache 2.0
SmolLM2-1.7B-Instruct	1.7	–	Apache 2.0
Ministral-3-3B-Instruct-2512	3	–	Apache 2.0
SmolLM3-3B	3	–	Apache 2.0
Qwen2.5-3B-Instruct	3	–	Qwen Research
Qwen3-4B-Instruct-2507	4	–	Apache 2.0
gemma-3-4b-it	4	–	Gemma
Trinity-Nano-Preview	6	1	Apache 2.0
Qwen3-8B	8	–	Apache 2.0
Ministral-3-8B-Instruct-2512	8	–	Apache 2.0
EuroLLM-9B-Instruct-2512	9	–	Apache 2.0
Ministral-3-14B-Instruct-2512	14	–	Apache 2.0
EuroLLM-22B-Instruct-2512	22	–	Apache 2.0
Mistral-Small-24B-Instruct-2501	24	–	Apache 2.0
Trinity-Mini	26	3	Apache 2.0
Qwen3-30B-A3B-Instruct-2507	30	3	Apache 2.0
Qwen3-Omni-30B-A3B-Instruct	30	3	Apache 2.0
Qwen3-32B	32	–	Apache 2.0
Qwen3-Next-80B-A3B-Instruct	80	3	Apache 2.0
Mixtral-8x22B-Instruct-v0.1	141	39	Apache 2.0
Qwen3-235B-A22B-Instruct-2507	235	22	Apache 2.0
Qwen3.5-397B-A17B	397	17	Apache 2.0
Mistral-Large-3-675B-Instruct-2512	675	41	Apache 2.0

F LEARNED REWARD MODEL

The learned reward model is a Bayesian linear regression head on the frozen last-hidden-state features (2560-dimensional) of Skywork-Reward-V2-Qwen3-4B, extracted once for every candidate response in the pool. The pipeline proceeds in outer batches of prompts: the head scores every candidate of the batch, Algorithm 1 selects one pair per prompt with the head's estimated reward gap in place of Δ_s, the oracle scores the two selected responses, and the head's posterior is updated on the new scores before the next batch; the update takes seconds. ActiveDelta ranks candidate pairs by the head's posterior-mean gap; DeltaUCB, the pipeline's own acquisition function, by an optimistic gap.

Comparison with DeltaUCB. Paired by seed, ActiveDelta leads DeltaUCB by 2.43 ± 0.16 (SE; paired t-test over five seeds, $p < 0.001$).

Ranking errors. In the final outer batch of the seed-42 runs, the head orders 1.0% of the selected pairs the wrong way at $C = 3$ and $C = 4$, and 2.1% at $C = 5$. The oracle label corrects the orientation of each such pair; a ranking error can therefore reduce the oracle gap of the selected pair but cannot introduce a wrong label.

Threshold. As with oracle ranking (Appendix B), raising C from 3 to 5 under the head lowers BBH and raises AlpacaEval.

G ONLINE DPO SETUP

The policy and the reference model are initialized from Olmo-3-7B-Instruct-SFT, and training is implemented in verl.[2] Each step draws 64 prompts from the 60k prompts of Section 6.1. Prompts longer than 2048 tokens are removed, which leaves an epoch of 926 steps, and responses are limited to 2048 tokens. Training uses AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay 0.01, a linear learning-rate schedule with 10% warmup, and gradients clipped at norm 20. Each run uses three nodes with four NVIDIA GH200 GPUs each: one generates the rollouts with SGLang, and two train the policy. The best configuration takes 10 hours. We tuned online DPO over 25 runs, one seed each: learning rates from 5×10^{-7} to 8×10^{-6} and β from 2.5×10^{-4} to 0.1, with responses sampled at temperature 1.0 and top-p 1.0, or at the defaults of Olmo 3, temperature 0.6 and top-p 0.95. The best configuration is $\beta = 0.001$, lr 2×10^{-6} and temperature 0.6.

Margin and trajectory. At the best configuration, the difference of policy log-ratios between the chosen and rejected responses reaches 11.3 nats at the end of the epoch and is still growing, against about 300 for offline MaxMin (Appendix I); at $\beta = 0.001$, this is an implicit margin of 0.011 and a gradient weight of 0.50. The seven-benchmark mean rises from 66.2 to 67.9, 70.3 and 71.2 at one third, two thirds and the end of the epoch, and the mean AlpacaEval response grows from 241 to 312 words. Online DPO prefers a much smaller β than offline training: at lr 2×10^{-6}, the mean rises by about 3.5 points from $\beta = 0.1$ to $\beta = 0.001$ and is flat below it. Learning rates above 2×10^{-6} degrade sharply: at $\beta = 0.025$, the mean falls from 68.9 at 2×10^{-6} to 64.8 at 6×10^{-6}.

H SECOND BASE MODEL

Tables 16 and 17 give the grid behind Section 6.5: Apertus-1.5-8B-SFT on the 60k prompts of Section 6.1, effective batch 32, one seed (42) per cell, 28 cells for MaxMin and 27 for ActiveDelta. MaxMin is best at lr 6×10^{-6} and $\beta = 0.05$, where both methods were trained with four further seeds on the same data order (Table 6); paired by seed, ActiveDelta leads by 0.9 ± 0.2 (SE; paired t-test over five seeds, $p \approx 0.01$).

With its best threshold, ActiveDelta is ahead of MaxMin at all nine settings of learning rate and β run for both methods, and averaged over the three thresholds at eight of them. No threshold is best across settings. ActiveDelta's BBH stays between 68 and 70 up to lr 4×10^{-6} and falls to

[2] https://github.com/volcengine/verl

about 65 at 8×10^{-6}. MaxMin trades BBH for AlpacaEval as β decreases: at $\beta = 0.025$ and lr 6×10^{-6}, it reaches the highest AlpacaEval score of all runs, 72.2, with BBH at 54.0. At the seeded configuration, ActiveDelta's AlpacaEval responses are longer than MaxMin's, 503 against 399 words.

Table 16: **MaxMin grid on Apertus-1.5-8B-SFT.** Seven-benchmark mean, 60k prompts, effective batch 32, seed 42.

Learning rate	$\beta = 0.025$	$\beta = 0.05$	$\beta = 0.1$	$\beta = 0.2$
5×10^{-7}	70.13	69.17	67.60	67.50
1×10^{-6}	71.16	69.76	69.14	68.26
2×10^{-6}	71.19	71.19	69.73	68.13
4×10^{-6}	70.39	71.40	70.31	69.40
5×10^{-6}	70.82	71.29	70.69	69.51
6×10^{-6}	70.50	**72.05**	70.93	70.03
8×10^{-6}	71.36	71.60	71.42	69.31

Table 17: **Threshold grid on Apertus-1.5-8B-SFT.** Seven-benchmark mean, 60k prompts, effective batch 32, seed 42. Each row is one setting of learning rate and β; the final column is MaxMin at the same setting, and bold marks the ActiveDelta cells above it. Five-seed means for the lr 6×10^{-6}, $\beta = 0.05$ row are in Table 6.

β	**Learning rate**	$C = 3$	$C = 4$	$C = 5$	**MaxMin**
0.05	1×10^{-6}	69.72	**70.61**	**70.68**	69.76
	2×10^{-6}	71.15	**72.21**	**71.26**	71.19
	4×10^{-6}	**71.95**	**72.37**	**72.26**	71.40
	5×10^{-6}	**71.98**	**72.51**	**73.09**	71.29
	6×10^{-6}	71.33	**72.70**	**72.42**	72.05
	8×10^{-6}	71.38	71.40	**72.42**	71.60
0.1	1×10^{-6}	**69.41**	**69.20**	**70.04**	69.14
	2×10^{-6}	**69.92**	**70.50**	**70.13**	69.73
	4×10^{-6}	**70.49**	69.92	**70.38**	70.31

I TRAINING DYNAMICS ACROSS THRESHOLDS

Table 18: Batch-mean training statistics over the final steps of the epoch on 60k prompts, with the gradient weight evaluated at the batch-mean margin; mean $\pm$ 95% CI over five seeds.

	Implicit margin	Gradient weight	Gradient norm	Pairs with $m_\theta > 0$
MaxMin	15.08 ± 0.19	3.1×10^{-7}	0.79 ± 0.09	99.7%
ActiveDelta $C = 3$	1.96 ± 0.02	0.124	11.5 ± 0.3	94.5%
ActiveDelta $C = 4$	2.80 ± 0.02	0.057	8.4 ± 0.4	96.8%
ActiveDelta $C = 5$	3.64 ± 0.04	0.026	6.7 ± 0.6	97.7%
ActiveDelta $C = 6$	4.49 ± 0.04	0.011	5.2 ± 0.5	98.3%
ActiveDelta $C = 7$	5.40 ± 0.03	0.0045	3.6 ± 0.5	98.9%

MaxMin's gradient norm falls from 20 to below 1, while ActiveDelta's ends between 3.6 and 11.5; the magnitude of $\nabla_\theta m_\theta$ therefore does not offset the collapse of the weight. The one-sided constraint also admits pairs the policy ranks the wrong way: 3–6% of selected pairs at $C \leq 4$, against 0.3% for MaxMin. MaxMin drives the difference of policy log-ratios between chosen and rejected responses to about 300 nats by the end of the epoch; ActiveDelta holds it between 40 and 110.

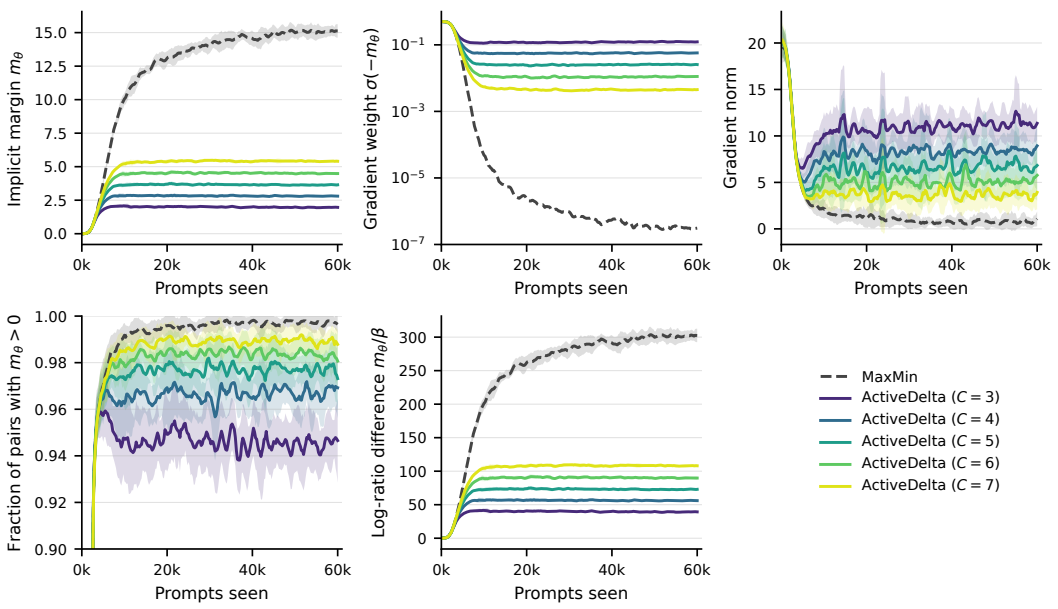

Figure 4: Training dynamics on 60k prompts for MaxMin (dashed) and every threshold, all with $\beta = 0.05$, lr 2×10^{-6} and effective batch 32: batch-mean implicit margin, gradient weight at this margin (log scale), gradient norm, fraction of selected pairs the policy ranks correctly, and the difference of policy log-ratios m_θ/β. Bands are 95% confidence intervals over five seeds.

J Evaluation harness and settings

All benchmarks are run with a fork of the Language Model Evaluation Harness (Gao et al., 2024) at version 0.4.13.dev0, with the vLLM backend and the settings of Table 19. Every prompt is formatted with the chat template of the evaluated model, and few-shot examples are given as preceding conversation turns. AlpacaEval 2 compares each response with the GPT-4 Turbo reference output of the benchmark and uses Llama-3.3-70B-Instruct as the judge; its win rates are therefore not comparable to those of the public leaderboard, which uses GPT-4 Turbo as the judge.

Table 19: **Evaluation settings.** HumanEval samples 20 completions per problem at temperature 0.8 and top-p 0.95; all other generation is greedy. The TruthfulQA prompt begins with the six question-answer examples of Lin et al. (2022).

Benchmark	Shots	Format	Metric
GSM8K	8	chain-of-thought generation	exact match
HumanEval	0	instruction-formatted generation	pass@10
BBH	3	chain-of-thought generation	exact match
IFEval	0	generation	prompt-level loose accuracy
AlpacaEval 2	0	generation, pairwise judge	length-controlled win rate
TruthfulQA	0	log-likelihood of the answer choices	MC2
BBQ	0	log-likelihood of the answer choices	accuracy