META-PAP: Meta-learning for Prompt-aware Preference Pairing in LLM Alignment

Pinlong Zhao[1] **Shiyu Hu**[2] **Jing Zhang**[3] **Mengyang Li**[4*]
[1]School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China
[2]School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore
[3]The Key Laboratory of Cognition and Decision Intelligence for Complex Systems,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
[4]Tianjin Key Laboratory of Wireless Mobile Communications and Power Transmission,
Tianjin Normal University, Tianjin, China

Abstract

Direct Preference Optimization (DPO) has become a popular route for aligning large language models with human preferences, but existing pipelines treat all preference pairs uniformly and ignore prompt-level heterogeneity. We show that the optimal pair-selection strategy varies systematically with prompt characteristics: prompts with concentrated reward distributions benefit from large reward-gap pairs that provide unambiguous signal, while prompts with diverse responses benefit from moderate-gap pairs because extreme positions are dominated by outliers. No fixed strategy is optimal across this heterogeneity. We propose META-PAP (Meta-learning for Prompt-aware Preference Pairing), a framework that turns this observation into a learned algorithm. META-PAP generates multiple diverse pairs per prompt via a structured grid over the response reward distribution, extracts a compact prompt-aware feature vector, and trains a lightweight meta-network through bilevel optimization to predict per-pair weights. The inner loop is a weighted DPO update; the outer loop evaluates a virtual policy on a small clean meta-set. Across Llama-3-8B, Mistral-7B-v0.3, and Llama-2-7B, META-PAP consistently outperforms vanilla DPO, fixed-position pairing, reward-gap weighting, top-k filtering, curriculum, and statistical rejection sampling, with average gains of $+4.1$ points on AlpacaEval 2.0 and $+3.6$ points on Arena-Hard, and approaches an oracle that exhaustively searches per prompt. The learned weighting policy is interpretable: it favors large-gap pairs on low-diversity prompts and softly rejects extreme pairs on high-diversity prompts. We additionally find that the meta-network requires only $M = 150$ pairs to saturate, transfers across domains, tolerates label noise on the meta-set, and remains effective with as few as $n = 32$ on-policy samples per prompt.

1 Introduction

Aligning large language models (LLMs) with human preferences has become a central component of post-training [4, 24]. Reinforcement Learning from Human Feedback (RLHF) [27, 29] typically learns a reward model from pairwise preferences and then optimizes the policy with reinforcement learning. Direct Preference Optimization (DPO) [25] simplifies this pipeline by directly optimizing a policy from preference pairs, avoiding explicit reward-model optimization during policy training.

*Corresponding author: limengyang@tjnu.edu.cn.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

This simplicity has made DPO-style objectives a widely used foundation for preference-based alignment [2, 10, 14, 23].

While the loss function has been extensively studied, the construction of the preference pairs themselves remains comparatively under-examined. Given a prompt and a pool of candidate responses, most pipelines rely on a fixed pairing rule: for example, selecting the highest- and lowest-scoring responses, filtering by reward margin, or applying a global curriculum over pair difficulty. Such rules are attractive because they are simple and easy to deploy, but they are also prompt-agnostic. They assume that the same kind of pair is informative for every prompt, regardless of how the model's responses are distributed.

This assumption is often violated. We find that the informativeness of a preference pair depends strongly on the prompt-level response distribution. For prompts whose candidate responses have concentrated reward distributions, large reward-gap pairs tend to provide clear supervision: the reward contrast can correspond to a meaningful quality difference. For prompts with diverse response distributions, however, extreme high- and low-reward responses are more likely to lie in sparse tails, where reward-model errors, length artifacts, or one-off generations can dominate. In this regime, moderate contrasts near the distributional bulk can provide a more stable learning signal. As a result, no single fixed pairing rule is uniformly effective across prompts.

This observation suggests that preference-pair construction should be adaptive. A useful pairing strategy should preserve decisive large-gap supervision when it is reliable, while softly downweighting extreme or noisy pairs when the prompt's response distribution is broad. Hand-designed heuristics struggle to capture this interaction: reward gap, distributional position, prompt-level diversity, and training stage jointly determine whether a pair is useful. Comparing models trained with different grid cells can provide a diagnostic reference, but training a separate model for each cell is costly.

We introduce META-PAP, a meta-learning framework for prompt-aware preference pairing. Instead of committing to a single pair per prompt, META-PAP samples multiple on-policy responses, scores them with a reward model, and constructs a structured set of candidate pairs over standardized reward positions. Each pair is represented by a compact feature vector capturing its reward gap, chosen and rejected positions, prompt-level response diversity, and training progress. A lightweight meta-network predicts a weight for each pair. The policy is updated with weighted DPO, while the meta-network is trained through a bilevel objective that evaluates a virtual policy update on a small high-confidence meta-set. In this way, META-PAP learns which kinds of pairs improve high-confidence held-out preferences, rather than relying on a fixed global rule.

Our experiments show that prompt-aware pairing consistently improves preference optimization. Across Llama-3-8B, Mistral-7B-v0.3, and Llama-2-7B, META-PAP outperforms vanilla DPO and strong pairing baselines, including fixed-position pairing, reward-gap weighting, filtering, curriculum, and statistical rejection sampling, on AlpacaEval 2.0, Arena-Hard, MT-Bench, and WildBench. Ablations show that the gains come from both the structured candidate-pair grid and the learned weighting rule, and diversity-stratified analysis confirms that META-PAP adapts its behavior across prompt regimes. The learned policy is interpretable: it favors large-gap pairs for low-diversity prompts and downweights unreliable extremes for high-diversity prompts.

Our contributions are summarized as follows:

- We identify prompt-level response heterogeneity as a key factor in preference-pair construction, and show that the best pairing strategy varies systematically with the reward distribution induced by each prompt.
- We propose META-PAP, a prompt-aware pair-weighting framework that combines structured candidate-pair generation with a lightweight meta-network trained by bilevel optimization.
- We demonstrate consistent improvements over DPO and strong pair-selection baselines across multiple model families and evaluation benchmarks, and show that the learned weighting policy is interpretable and practical.

2 Related Work

Preference optimization and pair construction. RLHF [24, 29] trains a reward model from pairwise preferences and optimizes the policy with PPO [27], while DPO [25] directly optimizes

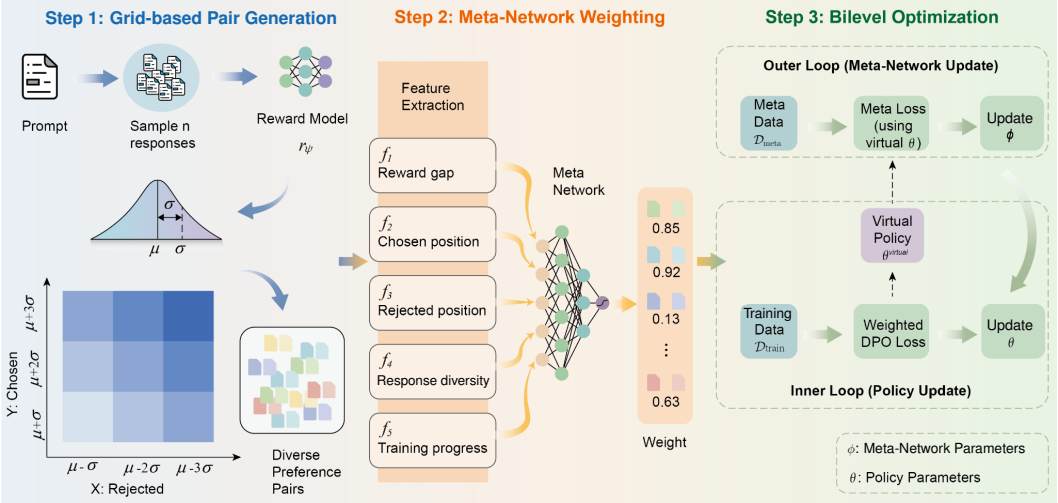


Figure 1: **Overview of META-PAP.** (Left) For each prompt, we sample n responses from the current policy and score them with a reward model to obtain a per-prompt reward distribution. (Middle) We generate diverse pairs via a 3×3 grid over the standardized reward space and extract a compact 5-dim prompt-aware feature for each pair. A lightweight meta-network predicts a weight from these features. (Right) The policy is trained via weighted DPO in the inner loop, and the meta-network is updated in the outer loop by evaluating a virtual policy on a small high-confidence meta-set, yielding a bilevel optimization that tailors the weighting to each prompt.

the policy from preference pairs. Subsequent variants, including IPO [2], ORPO [14], SimPO [23], KTO [10], and RSO [21], refine the objective or address distribution mismatch. These methods mainly specify how a given pair should update the policy; META-PAP instead studies which candidate pairs should be trusted, learning prompt-conditioned weights over pairs.

Data selection, curricula, and meta-reweighting. Data selection and curriculum learning filter or order examples by quality, influence, confidence, or difficulty [5, 6, 16, 22, 33, 37]. Preference-learning analogues such as reward-margin filtering and curriculum-by-gap impose fixed global rules over pair difficulty [21, 35]. TAPC [17] learns task-conditioned confidence targets from perplexity signals. Meta-reweighting methods such as L2RW [26] and MW-Net [28] learn example weights from validation feedback, but are typically designed for single-example supervised losses. META-PAP brings this principle to DPO by weighting preference pairs with prompt-aware distributional features and a small automatically constructed meta-set.

3 The META-PAP Framework

We first present empirical motivation for prompt-conditioned weighting (Sec. 3.1), formalize the weighted DPO objective (Sec. 3.2), describe the grid-based pair generation and feature design (Sec. 3.3–3.4), and finally specify the bilevel optimization that trains the meta-network (Sec. 3.5). Figure 1 shows the full pipeline.

3.1 Empirical motivation: response diversity controls optimal pairing

We sample $n = 200$ responses per prompt with Llama-2-7B-Chat on Anthropic-Helpful [3], score them with the reward model, and stratify prompts by response diversity $\sigma_x = \text{std}(\{r_\psi(x, y_j)\}_{j=1}^n)$. The strata contain 4,218 (38.3%) low-diversity prompts, 4,891 (44.4%) medium-diversity, and 1,901 (17.3%) high-diversity; we verify in Appendix C.5 that the cross-over pattern holds across reasonable threshold choices. For each stratum we train DPO on three fixed pairing rules: small-gap $(\mu_x+\sigma_x, \mu_x-\sigma_x)$, medium-gap $(\mu_x+2\sigma_x, \mu_x-2\sigma_x)$, and large-gap $(\mu_x+3\sigma_x, \mu_x-3\sigma_x)$. Table 1 reports the best- and worst-performing gap rules in each stratum and their win-rate difference (AlpacaEval 2.0 length-controlled win rate; full per-cell numbers in Appendix C.5).

Table 1: **Optimal gap depends on prompt-level response diversity**, on Llama-2-7B-Chat with AlpacaEval 2.0. The same large-gap rule that is best on low-diversity prompts is the worst on high-diversity prompts.

Response diversity	Best gap	Worst gap	Win-rate Δ (points)
Low ($\sigma_x < 1.5$)	Large	Small	+12.3
Medium ($1.5 \leq \sigma_x \leq 3.0$)	Medium	Large	+6.8
High ($\sigma_x > 3.0$)	Small	Large	+8.7

The relationship inverts across strata: the best gap on low-diversity prompts is the *worst* gap on high-diversity prompts. One possible explanation is that reward contrasts differ in reliability across the response distributions induced by different prompts, with extreme responses more susceptible to reward-model errors or atypical generations [12]. Table 1 compares each gap rule in isolation; learned weights over a mixture of grid pairs can favor a different non-extreme gap (Appendix E.4). A static rule cannot accommodate both regimes, motivating learned adaptation.

3.2 Weighted DPO objective

Let $\mathcal{D}_{\text{train}} = \{x_i\}_{i=1}^{N}$ be the prompts, π_θ a policy initialized from supervised fine-tuning, π_{ref} a frozen reference, and r_ψ a reward model. The DPO loss for a pair (x, y_w, y_l) is

$$\mathcal{L}_{\text{DPO}}(\theta; x, y_w, y_l) = -\log \sigma(\beta\, h_\theta(x, y_w, y_l)), \quad h_\theta = \log \tfrac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \tfrac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}. \tag{1}$$

Our Vanilla DPO baseline uses a single pair per prompt. We instead generate K pairs per prompt with feature vectors $\{\mathbf{f}_k\}_{k=1}^{K}$ and introduce a meta-network $V_\phi : \mathbb{R}^5 \to [0, 1]$ that predicts per-pair weights. The weighted training objective is

$$\mathcal{L}_{\text{train}}(\theta; \phi) = \mathbb{E}_{x \sim \mathcal{D}_{\text{train}}} \left[\frac{1}{K} \sum_{k=1}^{K} V_\phi(\mathbf{f}_k) \cdot \mathcal{L}_{\text{DPO}}^{(k)}(\theta; x) \right]. \tag{2}$$

Setting $V_\phi \equiv 1$ recovers training on the full grid with uniform weights (a strong baseline we ablate in Sec. 4). META-PAP learns V_ϕ from prompt-aware features to reduce DPO loss on a held-out high-confidence meta-set.

3.3 Grid-based pair generation

For each prompt x, we sample n responses from the current policy at temperature $T = 0.8$, score them with r_ψ, and compute the empirical mean μ_x and standard deviation σ_x.[2] We then define a 3×3 grid over standardized reward positions:

$$\mathcal{P}_{\text{ch}} = \{\mu_x + \sigma_x,\ \mu_x + 2\sigma_x,\ \mu_x + 3\sigma_x\}, \quad \mathcal{P}_{\text{re}} = \{\mu_x - \sigma_x,\ \mu_x - 2\sigma_x,\ \mu_x - 3\sigma_x\}. \tag{3}$$

For each $(p_c, p_r) \in \mathcal{P}_{\text{ch}} \times \mathcal{P}_{\text{re}}$, we select responses closest to the target positions, $y_w^{(k)} = \arg\min_{y_j} |r_j - p_c|$ and $y_l^{(k)} = \arg\min_{y_j} |r_j - p_r|$, and discard duplicates. This yields at most nine unique pairs per prompt covering small, medium, and large gaps as well as asymmetric strategies. The grid is principled in three respects: (i) it spans qualitatively different pairing strategies without exhaustive search; (ii) all pairs share the same response sample, so generation cost is amortized; and (iii) it provides interpretable strategy coverage which Sec. 3.5 exploits via learned interpolation. Appendix A gives a coverage argument and Appendix D ablates grid resolution.

3.4 Prompt-aware feature design

For each pair $(y_w^{(k)}, y_l^{(k)})$ we extract a compact 5-dimensional feature vector

$$\mathbf{f}_k = \left[r_w^{(k)} - r_l^{(k)},\ \tfrac{r_w^{(k)} - \mu_x}{\sigma_x + \epsilon},\ \tfrac{r_l^{(k)} - \mu_x}{\sigma_x + \epsilon},\ \sigma_x,\ t/T_{\text{total}} \right]^\top, \tag{4}$$

[2] We use $n = 200$ for the main experiments and analyze sample efficiency in Sec. 4.5; $n = 32$ retains $\approx 99\%$ of the win rate.

Algorithm 1 META-PAP training step

Require: Prompt batch $\mathcal{B}_x$, meta batch $\mathcal{B}_{\text{meta}}$, policy π_θ, meta-network V_ϕ, learning rates α, η

1: **Data generation:** Sample n responses per prompt; form grid pairs $\mathcal{B}_{\text{gen}}$; extract features $\mathbf{F} = \{\mathbf{f}_k\}$.
2: **Virtual update:** $\mathbf{w} \leftarrow V_\phi(\mathbf{F})$; $\tilde{\theta}(\phi) \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\text{train}}(\theta; \phi)$.
3: **Meta update:** $\mathcal{L}_{\text{meta}}(\phi) \leftarrow \mathcal{L}_{\text{DPO}}(\tilde{\theta}(\phi); \mathcal{B}_{\text{meta}})$; $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}_{\text{meta}}$.
4: **Main update:** $\mathbf{w}' \leftarrow V_\phi(\mathbf{F})$; $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\text{train}}(\theta; \phi)$ using $\mathbf{w}'$.
5: **return** θ, ϕ

where $r_w^{(k)}, r_l^{(k)}$ are the rewards of the chosen and rejected responses, t is the current step, T_{total} is the total number of steps, and $\epsilon = 10^{-3}$ stabilizes normalization. The five components capture (i) discriminability via the reward gap; (ii) the standardized chosen and (iii) rejected positions relative to the distribution, exposing whether either response is in the tails; (iv) prompt-level diversity σ_x, the key signal from Sec. 3.1; and (v) training progress, allowing the policy to adapt the curriculum over time. Here, prompt awareness refers to each prompt's response-distribution statistics rather than semantic prompt embeddings. The features are deliberately kept low-dimensional so that the meta-network is small and generalizes from a small meta-set.

3.5 Bilevel optimization

We train V_ϕ so that the resulting weighted update minimizes loss on a small high-confidence meta-set $\mathcal{D}_{\text{meta}}$:

$$\min_{\phi} \; \mathcal{L}_{\text{meta}}(\phi) = \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}_{\text{meta}}} \left[\mathcal{L}_{\text{DPO}}(\theta^\star(\phi); x, y_w, y_l) \right], \quad \text{s.t. } \theta^\star(\phi) = \arg\min_{\theta} \mathcal{L}_{\text{train}}(\theta; \phi). \tag{5}$$

We approximate the converged inner solution with a one-step virtual update $\tilde{\theta}(\phi) = \theta - \alpha \nabla_\theta \mathcal{L}_{\text{train}}(\theta; \phi)$, following validation-guided meta-reweighting [26, 28]. For this local meta-update, the current policy parameters, sampled pairs, and reward features are held fixed, while differentiation is retained through the weights $V_\phi(\mathbf{f}_k)$ and hence through $\tilde{\theta}(\phi)$ (Appendix A.3). Each training step has three stages:

1. **Virtual update.** With current ϕ, compute $\mathbf{w} = V_\phi(\mathbf{F})$ and form $\tilde{\theta}(\phi) = \theta - \alpha \nabla_\theta \mathcal{L}_{\text{train}}(\theta; \phi)$.
2. **Meta update.** Evaluate $\mathcal{L}_{\text{meta}}$ at $\tilde{\theta}$ and update $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}_{\text{meta}}$. The gradient flows through the weight-dependent virtual update, using $\nabla_{\tilde{\theta}} \mathcal{L}_{\text{meta}}$ as the chain-rule vector.
3. **Main update.** Recompute $\mathbf{w}' = V_{\phi'}(\mathbf{F})$ and update $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\text{train}}(\theta; \phi')$.

Architecture. The meta-network is a 3-layer MLP $\text{Linear}(5, 64) \to \text{ReLU} \to \text{Linear}(64, 32) \to \text{ReLU} \to \text{Linear}(32, 1) \to \sigma$, totaling 2,497 parameters. Weights use Xavier uniform initialization and biases are zero, with sigmoid outputs intended to start near 0.5. This does not exactly recover the $w \equiv 1$ Uniform Grid baseline.

Meta-set construction. $\mathcal{D}_{\text{meta}}$ is constructed automatically without human annotation. We retain pairs with reward gap $\Delta r > 0.3$ and apply stratified sampling across response-diversity strata (low/medium/high), yielding a balanced set of $M = 150$ pairs covering the full heterogeneity spectrum. The meta-set is disjoint from training prompts and evaluation benchmarks; high-confidence filtering does not make its labels independent ground truth. Sec. 4.6 shows that performance saturates around $M = 150$, and Appendix D.8 compares construction methods including bootstrapping and human screening.

Appendix A derives the local gradient-alignment signal: aligned pairs receive an incentive for larger weights, while conflicting pairs receive the opposite incentive. Shared network parameters couple these updates; the analysis provides intuition rather than a guarantee for every weight or for convergence.

3.6 Discussion of design choices

A natural alternative to META-PAP is to threshold features directly: drop pairs with $|r_w - \mu_x| > 2\sigma_x$, weight by $\sigma(\Delta r)$, or schedule by training step. We compare related fixed rules—Reward-Gap Weighting, Top-50% Filtering, and Curriculum-by-Gap—in Sec. 4; they underperform META-PAP in our experiments. The five features in Eq. 4 form a compact, empirically effective representation of reward contrast, distributional position, diversity, and training stage. Additional higher-order features give limited improvement (Appendix E.2). The one-step meta-update avoids solving the full inner optimization problem at every meta-step; the small network reduces its parameter cost but does not establish the accuracy of this approximation. The default weights are unnormalized, so both relative pair contributions and gradient scale can change. Normalized comparisons in Sec. 4.7 retain gains over Uniform Grid. The main computational cost remains on-policy generation (Sec. 4.5); meta-network operations alone should not be equated with the cost of the entire bilevel update.

4 Experiments

4.1 Setup

Models, reward model, and datasets. We evaluate META-PAP on **Llama-3-8B** [1], **Mistral-7B-v0.3** [15], and **Llama-2-7B** [30]. All models are first supervised fine-tuned on Alpaca, then aligned with preference optimization. The reward model is **ArmoRM-Llama3-8B-v0.1** [31] for response scoring, meta-set construction, and feature extraction; we verify in Appendix D.6 that the gains are stable across three reward-model choices (ArmoRM, UltraRM-13B [8], Starling-RM-7B [38]). The primary training set is **Anthropic-Helpful** [3] (43,835 pairs); for cross-domain analysis we additionally train on **UltraFeedback** [8]. Grid generation produces an empirical mean of 7.6 ± 1.1 unique pairs per prompt after deduplication.

Benchmarks. We report on four benchmarks: **AlpacaEval 2.0** [9] (length-controlled win rate vs. GPT-4-Turbo), **Arena-Hard** [18], **MT-Bench** [36] (eight categories), and **WildBench** [20] (real-world queries). The main results report mean $\pm$ std over three seeds; supplementary point estimates are identified separately.

Baselines. We compare against (i) standard / static strategies—**Vanilla DPO** (max-min from $n = 200$ responses per prompt, matching the per-prompt sample count), and **Static-$\mu\pm2\sigma$**, the best fixed-position rule from Sec. 3.1; (ii) heuristic and selection methods—**Reward-Gap Weighting** (sigmoid of Δr), **Top-50% Filtering**, **Curriculum-by-Gap**, and **RSO** [21]; (iii) on-policy / iterative methods—**Online DPO** [13], **Iterative DPO** (3 iterations), and **SPIN** [7], which represent the natural class of methods that, like META-PAP, sample from the current policy during training; and (iv) component baselines—**Uniform Grid** (our grid with uniform weights, isolating the impact of the learned weighting) and **Oracle** (a diagnostic reference based on nine separately trained grid-cell models and meta-set loss; see Appendix C.3).

Implementation. We use AdamW with learning rate $\alpha = 5\times10^{-7}$ for Llama-2-7B and $\alpha = 1\times10^{-6}$ for Llama-3-8B and Mistral-7B. The meta-network uses $\eta = 1\times10^{-4}$. We train for one epoch with batch size 64, DPO temperature $\beta = 0.1$, and response sampling temperature $T = 0.8$, using bfloat16 precision on 8×A100 (80GB) GPUs. Full hyperparameters are in Appendix B.

4.2 Main results

Table 2 reports results across all three models and four benchmarks. META-PAP consistently outperforms the training baselines across all four metrics; Oracle serves as a diagnostic reference.

Gains over single-pair and heuristic baselines. Static and Vanilla DPO discard information by collapsing the response distribution to one contrast. META-PAP leverages the full grid: on Llama-3-8B it achieves 26.8% on AlpacaEval 2.0, +4.4 over Vanilla DPO and +3.0 over Static-$\mu\pm2\sigma$. The same pattern holds on Mistral-7B (+3.9) and Llama-2-7B (+3.9). Heuristics such as RSO and Reward-Gap Weighting improve over Vanilla DPO, but do not adapt their rule to prompt-level response diversity: META-PAP outperforms RSO by 2.5 on Llama-3-8B and by similar margins on

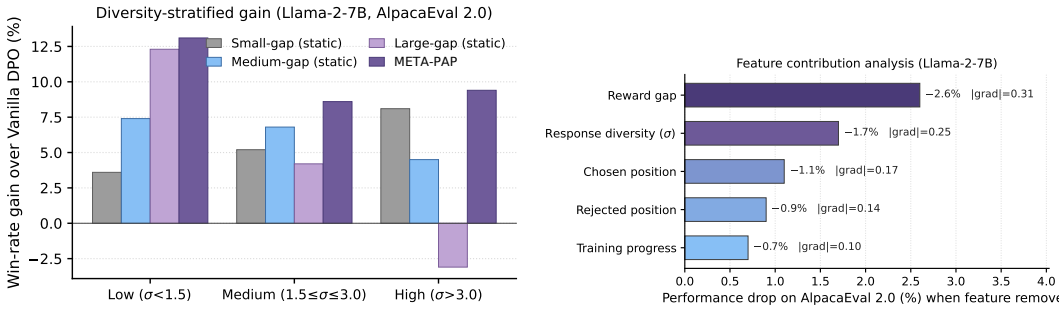
Table 2: **Main results across three base models and four benchmarks.** META-PAP outperforms all single-pair, heuristic, and uniform-grid baselines. Oracle denotes a diagnostic grid-cell reference, not a formal performance bound. Numbers are mean ± std over three seeds.

Model	Method	AlpacaEval 2.0	Arena Hard	MT-Bench	Wild-Bench	Pairs/ Prompt
Llama-3-8B	Vanilla DPO	22.4±0.4	18.3±0.3	7.85±0.06	41.2±0.5	1
	Static-$\mu\pm2\sigma$	23.8±0.4	19.5±0.3	7.98±0.06	42.8±0.5	1
	Reward-Gap Weighting	24.1±0.4	19.8±0.3	8.02±0.06	43.1±0.5	1
	Top-50% Filtering	23.5±0.4	19.2±0.3	7.94±0.06	42.5±0.5	1
	Curriculum-by-Gap	24.0±0.4	19.6±0.3	8.00±0.06	42.9±0.5	1
	RSO	24.3±0.4	20.0±0.3	8.05±0.06	43.4±0.5	1
	Uniform Grid	25.1±0.5	20.8±0.4	8.15±0.07	44.3±0.6	∼8
	META-PAP	**26.8±0.5**	**22.1±0.4**	**8.32±0.07**	**45.9±0.6**	∼8
	Oracle (reference)	28.1±0.5	23.4±0.4	8.48±0.07	47.2±0.6	∼8
Mistral-7B-v0.3	Vanilla DPO	19.6±0.4	15.2±0.3	7.54±0.06	38.7±0.5	1
	Static-$\mu\pm2\sigma$	20.8±0.4	16.3±0.3	7.68±0.06	40.1±0.5	1
	Reward-Gap Weighting	21.1±0.4	16.6±0.3	7.72±0.06	40.5±0.5	1
	Top-50% Filtering	20.5±0.4	16.0±0.3	7.64±0.06	39.8±0.5	1
	Curriculum-by-Gap	20.9±0.4	16.4±0.3	7.70±0.06	40.3±0.5	1
	RSO	21.3±0.4	16.8±0.3	7.75±0.06	40.8±0.5	1
	Uniform Grid	22.0±0.5	17.5±0.4	7.86±0.07	41.7±0.6	∼8
	META-PAP	**23.5±0.5**	**18.9±0.4**	**8.01±0.07**	**43.2±0.6**	∼8
	Oracle (reference)	24.7±0.5	20.1±0.4	8.16±0.07	44.5±0.6	∼8
Llama-2-7B	Vanilla DPO	13.2±0.3	8.7±0.2	6.42±0.05	32.4±0.4	1
	Static-$\mu\pm2\sigma$	14.6±0.4	9.4±0.3	6.58±0.06	33.8±0.5	1
	Reward-Gap Weighting	14.9±0.3	9.7±0.3	6.63±0.05	34.2±0.5	1
	Top-50% Filtering	14.4±0.4	9.2±0.3	6.54±0.06	33.5±0.5	1
	Curriculum-by-Gap	14.7±0.3	9.5±0.2	6.61±0.05	33.9±0.5	1
	RSO	15.1±0.4	9.9±0.3	6.68±0.06	34.6±0.5	1
	Uniform Grid	15.6±0.4	10.4±0.3	6.78±0.06	35.4±0.5	∼8
	META-PAP	**17.1±0.4**	**11.9±0.3**	**7.08±0.06**	**37.2±0.5**	∼8
	Oracle (reference)	18.2±0.5	12.8±0.4	7.21±0.07	38.5±0.6	∼8

the other two models, supporting the benefit of learned prompt-conditioned weighting over the tested hand-designed rules.

Disentangling augmentation from reweighting. Whether META-PAP's gains come merely from training on a richer set of pairs is examined by the Uniform Grid baseline. META-PAP beats it by $+1.7$ on AlpacaEval 2.0 and $+1.3$ on Arena-Hard for Llama-3-8B, providing an additional improvement beyond uniform grid training. On Llama-2-7B, comparisons using the same candidate grid give $17.1/11.9$ for META-PAP versus $15.6/10.4$ for Uniform Grid and $15.3/10.2$ for reward-gap weighting (AlpacaEval/Arena-Hard; Appendix D.11). These controls isolate the benefit of weighting among multiple pairs more directly than comparisons to single-pair baselines.

On-policy / iterative methods and reward-model robustness. Because META-PAP samples from the current policy at training time, the natural reference class is on-policy and iterative DPO. On Llama-2-7B, META-PAP (17.1 AlpacaEval / 11.9 Arena-Hard) outperforms Online DPO [13] (15.4/10.1), 3-iteration Iterative DPO (15.8/10.4), and SPIN [7] (14.9/9.8). The two axes are orthogonal: those methods address *data freshness*, whereas META-PAP addresses *within-batch pair quality*, and they could be combined (Appendix D.7). The gains are also stable when replacing the reward model throughout the pipeline. More directly, fixing ArmoRM for training-side scoring and using UltraRM-13B for meta-supervision yields 17.0 AlpacaEval and 11.7 Arena-Hard, close to the default $17.1/11.9$ (Appendix D.6). This supports robustness to meta-supervision choice without establishing immunity to shared reward-model biases.



(a) Win-rate gain by response-diversity stratum.

(b) Feature contribution analysis on Llama-2-7B.

Figure 2: **Where the gains come from.** (a) Static gap rules each excel on a single diversity stratum and fail on others; META-PAP matches or exceeds the best static rule on every stratum simultaneously. (b) Removing the reward gap and response diversity features causes the largest performance drops, confirming the prompt-conditional mechanism identified in Sec. 3.1.

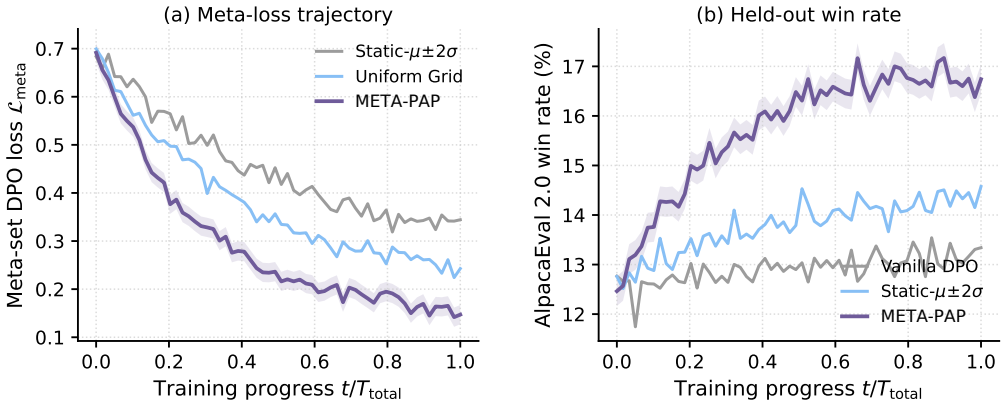


Figure 3: **Training dynamics.** (a) The bilevel meta-loss decreases overall under META-PAP and generally remains below the Static and Uniform-Grid trajectories, indicating that the learned weights actively reduce error on held-out high-confidence preferences. (b) Held-out AlpacaEval 2.0 win rate diverges early in training and the gap between META-PAP and the static baseline remains substantial, consistent with the emergent decisive-weighting behavior in Sec. 4.6.

4.3 Diversity-stratified gain and feature ablation

Figure 2a confirms the prompt-conditional mechanism: each static gap rule wins on one stratum and loses on the others (large-gap is best on low-diversity but suffers -3.1 on high-diversity, while small-gap is best on high-diversity but only marginally helpful on low-diversity), whereas META-PAP matches or exceeds the best static rule on every stratum simultaneously. Figure 2b validates the feature design: removing the reward gap causes the largest drop (-2.6), followed by response diversity (-1.7); these separate ablations identify reward gap and response diversity as influential inputs, with smaller drops for position features and training progress. These leave-one-out ablations measure individual removals rather than an additive decomposition of the total improvement.

4.4 Training dynamics

Figure 3 shows two views of the training process. The meta-loss decreases overall for META-PAP. On held-out AlpacaEval 2.0 the gap to the static baseline opens early and remains substantial, although the trajectories fluctuate.

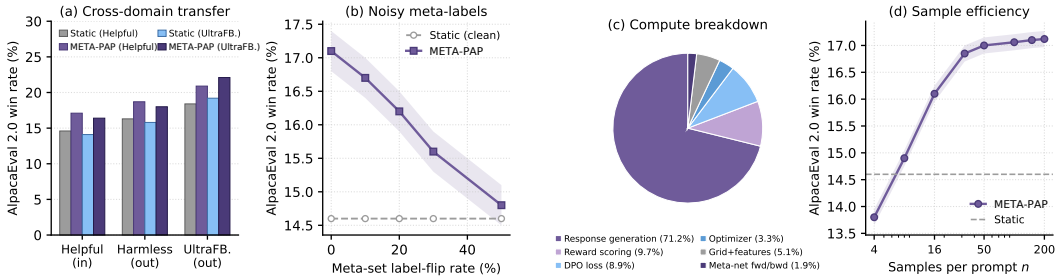


Figure 4: **Robustness, compute, and sample efficiency on Llama-2-7B.** (a) Cross-domain transfer: gains remain positive across the reported domains. (b) At 20% meta-label flips META-PAP still beats the static baseline by 1.6 points; at 50% label flips its score approaches the static baseline. (c) Per-epoch compute is dominated by response generation (71%); meta-network operations are <2%. (d) Using $n = 32$ samples retains ≈99% of the win rate of $n = 200$, using 84% fewer response samples; the reported practical overhead is ∼15%.

4.5 Robustness, sample efficiency, and compute cost

The bilevel framework relies on the quality of the meta-set. Figure 4(a) shows cross-domain transfer: training on Helpful and evaluating on Harmless or UltraFeedback (out-of-domain), META-PAP retains ∼95% of its in-domain advantage, indicating the learned weighting captures distributional structure rather than domain-specific content. Figure 4(b) probes meta-label noise: at 20% flips META-PAP still beats the static baseline by 1.6 points, with graceful degradation up to 50% flips where performance approaches the static baseline. Figure 4(c) decomposes the per-epoch cost: generation accounts for 71% and meta-network operations alone for less than 2%. These components do not isolate the full cost of the virtual policy and meta-loss evaluation. Figure 4(d) shows that $n = 32$ retains ≈99% of the $n = 200$ win rate while using 84% fewer response samples; the reported practical overhead is ∼15%.

4.6 Analysis: what does the meta-network learn?



(a) Meta-set efficiency (b) Learned weighting policy (c) Weight distribution over time

Figure 5: **Meta-network behavior.** (a) Performance saturates by $M = 150$. (b) Large gaps are favored when σ_x is small; moderate gaps are favored when σ_x is large, recovering the diversity-dependent pattern while favoring a mixture rather than one isolated gap rule. (c) Weight distribution evolves from unimodal at 0.5 to bimodal, an emergent curriculum.

Adaptive signal-to-noise control. Figure 5b shows that the meta-network assigns different weights across response-diversity regimes. For low-diversity prompts, the network assigns large-gap pairs weights $w > 0.8$, consistent with the empirical advantage of large-gap rules in Table 1. For high-diversity prompts, the network suppresses extreme pairs ($w < 0.4$) and gives more weight to medium-gap than small-gap pairs, consistent with a preference for less extreme contrasts. Table 1 evaluates gap rules in isolation, whereas the heatmap summarizes weights over a mixture of pairs; the two comparisons need not select the same gap. The observed weights alone do not establish why a particular pair is useful (Appendix E.4).

Emergent decisive curriculum. Figure 5c shows the temporal evolution of weights. They cluster unimodally near 0.5 early in training and become increasingly bimodal as training proceeds, assigning some pairs weights near 1 and others weights near 0. This shows increasingly selective weighting; the weight distribution alone does not identify which pairs are noisy. The curriculum emerges from the bilevel objective without an explicit schedule.

Scalable meta-supervision. Figure 5a shows that performance saturates by $M = 150$ meta-pairs and already provides substantial improvement at $M = 75$, showing that a small meta-set is sufficient in this experimental setting. Appendix D.8 compares high-confidence filtering with human screening of candidate meta-pairs.

4.7 Relative pair weights and optimization scale

The sigmoid weights in Eq. 2 affect relative pair contributions and overall gradient scale. Table 3 controls the candidate grid while comparing default weights with fixed per-prompt mean, fixed per-prompt sum, and softmax normalization.

Table 3: Weight-normalization comparison on Llama-2-7B with the same candidate grid. Results are point estimates without uncertainty estimates.

Normalization	AlpacaEval 2.0	Arena-Hard
Unnormalized (default)	17.1	11.9
Per-prompt mean $= 0.5$	16.8	11.6
Per-prompt sum $= K$	16.7	11.5
Per-prompt softmax	16.6	11.4
Uniform Grid	15.6	10.4

The normalized variants obtain 16.6–16.8 on AlpacaEval 2.0 and 11.4–11.6 on Arena-Hard, compared with $15.6/10.4$ for Uniform Grid. These gains support a role for relative pair weighting beyond the change in average weight. The default reaches $17.1/11.9$.

Shared-grid controls (Appendix D.11) give $15.3/10.2$ for reward-gap weighting, below Uniform Grid and learned weighting. The weighting rule therefore matters beyond merely using unequal weights. Best-of-grid and random-from-grid optimize one pair per prompt, so those comparisons also change the amount of pair-level supervision.

5 Discussion and Conclusion

This work argues that preference-pair construction is not a fixed preprocessing choice, but a prompt-conditioned learning problem. We showed that pair informativeness depends on the response distribution induced by each prompt: large-gap pairs provide decisive supervision when rewards are concentrated, while extreme tail pairs on high-diversity prompts are more susceptible to reward-model artifacts and outlier generations [12]. META-PAP turns this observation into a learned algorithm by constructing a structured grid of candidate pairs and using a lightweight meta-network, trained through bilevel optimization, to predict prompt-aware pair weights. Across three model families and four benchmarks, META-PAP consistently improves over DPO and strong heuristic baselines and learns an interpretable weighting policy that favors decisive contrasts on low-diversity prompts while downweighting unreliable extremes on high-diversity prompts. These results suggest that effective preference optimization depends not only on the objective, but also on how preference evidence is constructed from model generations. Limitations remain: META-PAP depends on a proxy reward model, summarizes response distributions with standardized reward statistics, and adds on-policy sampling cost. Future work could explore stronger meta-supervision, adaptive grids for multimodal distributions, and integration with online or verifier-based alignment.

Acknowledgments and Disclosure of Funding

The authors received no funding for this work and declare no competing interests.

References

[1] AI@Meta. Llama 3 model card, 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[2] Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Rémi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *AISTATS*, 2024.

[3] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[4] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, pages 41–48, 2009.

[6] Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023.

[7] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In *International Conference on Machine Learning*, 2024.

[8] 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 *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[9] Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

[10] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

[11] Simin Fan, Matteo Pagliardini, and Martin Jaggi. DoGE: Domain reweighting with generalization estimation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 12895–12915, 2024.

[12] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866, 2023.

[13] 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. *arXiv preprint arXiv:2402.04792*, 2024.

[14] Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691*, 2024.

[15] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[16] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, volume 23, 2010.

[17] Mengyang Li, Zhong Zhang, and Pinlong Zhao. Task-aware preference calibration for direct preference optimization. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 72616–72641. PMLR, 2026. URL https://proceedings.mlr.press/v306/li26jp.html.

[18] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. In *Forty-second International Conference on Machine Learning*, 2025.

[19] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[20] Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *arXiv preprint arXiv:2406.04770*, 2024.

[21] Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, and Jialu Liu. Statistical rejection sampling improves preference optimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=xbjSwwrQOe.

[22] Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *ICLR*, 2024.

[23] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*, 2024.

[24] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[25] 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. *Advances in Neural Information Processing Systems*, 36, 2023.

[26] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343, 2018.

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

[28] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[29] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[30] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[31] Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. *arXiv preprint arXiv:2406.12845*, 2024.

[32] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2024.

[33] Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

[34] Kailai Yang, Zhiwei Liu, Qianqian Xie, Jimin Huang, Erxue Min, and Sophia Ananiadou. Selective preference optimization via token-level reward function estimation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025. URL https://aclanthology.org/2025.emnlp-main.359/.

[35] Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. Rrhf: Rank responses to align language models with human feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 10935–10950, 2023.

[36] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2023.

[37] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36, 2023.

[38] Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, Karthik Ganesan, Wei-Lin Chiang, Jian Zhang, and Jiantao Jiao. Starling-7B: Improving helpfulness and harmlessness with RLAIF. In *First Conference on Language Modeling*, 2024.

Appendix Contents

A Theoretical Perspective **15**
- A.1 Bilevel formulation and notation 15
- A.2 First-order pair-gradient alignment 15
- A.3 One-step meta-gradient approximation 16
- A.4 Grid coverage of the strategy space 16
- A.5 Optimization landscape and convergence 16
- A.6 Discussion on distribution assumptions 16

B Architecture and Implementation Details **17**
- B.1 Complete hyperparameter configuration 17
- B.2 Meta-network architecture 17
- B.3 Computational infrastructure 17
- B.4 Engineering considerations 18

C Data, Metrics, and Baselines **18**
- C.1 Data splits and reward-model details 18
- C.2 Metric definitions 18
- C.3 Baseline configurations 18
- C.4 Mean and standard-deviation reporting 19
- C.5 Motivation experiment details 19

D Additional Experimental Results **20**
- D.1 Cross-domain generalization 20
- D.2 Noise tolerance 20
- D.3 Response-distribution statistics 20
- D.4 Extended training 20
- D.5 Scaling analysis 21
- D.6 Reward-model sensitivity 21
- D.7 Comparison with on-policy and iterative DPO baselines 21
- D.8 Meta-dataset construction comparison 22
- D.9 Computational breakdown 22
- D.10 Hyperparameter sensitivity 23
- D.11 Comparisons using a shared candidate grid 23
- D.12 Weight normalization 23

E Qualitative Analysis **24**
- E.1 Case studies 24
- E.2 Feature importance via gradients 24
- E.3 Failure cases 24
- E.4 Isolated gap rules and learned mixtures 25

F Sample-Efficiency Analysis **25**

G Extended Related Work **26**

H Limitations and Broader Impact **26**

A Theoretical Perspective

This appendix expands the meta-gradient argument used in the main paper. The analysis characterizes the local gradient-alignment signal of the one-step virtual update, linking pair weights to the high-confidence held-out objective. It does not establish globally optimal weights or a convergence guarantee. Figure 6 visualizes the same statement geometrically.

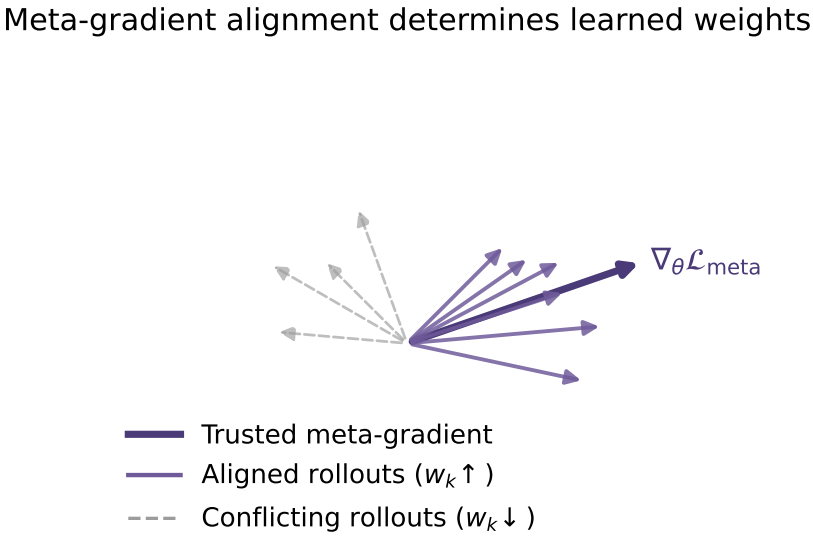


Figure 6: **Meta-gradient intuition.** The local analysis (§A.2) associates positive pair–meta gradient alignment with an incentive for greater weight and negative alignment with the opposite incentive. The arrows illustrate this local signal; shared meta-network parameters couple the realized changes in individual weights.

A.1 Bilevel formulation and notation

Let $\mathbf{F} = \{\mathbf{f}_k\}_{k=1}^{KB}$ denote the features of all KB pairs in the current batch (B prompts, K pairs per prompt) and $\mathbf{w} = V_\phi(\mathbf{F}) \in [0,1]^{KB}$ the corresponding meta-network outputs. With pair gradients $\mathbf{g}_k = \nabla_\theta \mathcal{L}_{\text{DPO}}^{(k)}(\theta)$, the weighted training gradient is $\mathbf{G}(\phi) = \frac{1}{KB}\sum_k V_\phi(\mathbf{f}_k)\,\mathbf{g}_k$. The virtual update is $\tilde{\theta}(\phi) = \theta - \alpha \mathbf{G}(\phi)$, the meta-loss is $\mathcal{L}_{\text{meta}}(\tilde{\theta}) = \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_{\text{meta}}}\mathcal{L}_{\text{DPO}}(\tilde{\theta}; x, y_w, y_l)$, and the meta-loss gradient with respect to the virtual policy is $\nabla_{\tilde{\theta}}\mathcal{L}_{\text{meta}}$. For clarity, we write the derivation for K pairs per prompt; with variable K_i, each prompt's terms are normalized by K_i before averaging over prompts.

A.2 First-order pair-gradient alignment

Lemma (alignment). For the one-step virtual update, holding the current policy parameters, sampled pairs, and reward features fixed,

$$\nabla_\phi \mathcal{L}_{\text{meta}}(\tilde{\theta}(\phi)) \;=\; -\frac{\alpha}{KB}\sum_{k=1}^{KB} \left\langle \mathbf{g}_k,\, \nabla_{\tilde{\theta}}\mathcal{L}_{\text{meta}} \right\rangle \nabla_\phi V_\phi(\mathbf{f}_k). \tag{6}$$

Proof sketch. By the chain rule, $\nabla_\phi \mathcal{L}_{\text{meta}} = \nabla_{\tilde{\theta}}\mathcal{L}_{\text{meta}}^\top \nabla_\phi \tilde{\theta}(\phi) = -\alpha\, \nabla_{\tilde{\theta}}\mathcal{L}_{\text{meta}}^\top \nabla_\phi \mathbf{G}(\phi)$. Substituting $\mathbf{G}(\phi) = \frac{1}{KB}\sum_k V_\phi(\mathbf{f}_k)\mathbf{g}_k$ and noting that $\mathbf{g}_k$ is independent of ϕ under these local assumptions yields Equation (6).

Implication. If the weights are viewed as independent coordinates, then

$$\frac{\partial \mathcal{L}_{\text{meta}}}{\partial w_k} = -\frac{\alpha}{KB}\langle \mathbf{g}_k, \nabla_{\tilde{\theta}}\mathcal{L}_{\text{meta}} \rangle. \tag{7}$$

Positive alignment therefore provides a local incentive to increase a pair's weight, whereas negative alignment provides the opposite incentive. With a shared meta-network, the weights are coupled through ϕ, so the sign of each individual weight change is not guaranteed. The observed downweighting of extreme pairs is consistent with this interpretation; it is not a consequence proved for every tail pair.

A.3 One-step meta-gradient approximation

At a differentiable stationary inner solution with an invertible Hessian, implicit differentiation involves

$$\frac{\partial \theta^\star}{\partial \phi} = -[\nabla_\theta^2 \mathcal{L}_{\text{train}}]^{-1} \nabla_\phi \nabla_\theta \mathcal{L}_{\text{train}}. \tag{8}$$

Instead of solving this sensitivity problem, we optimize the one-step surrogate defined by the virtual update. With current policy parameters and pair features held fixed, its meta-gradient is given by Eq. (6). The chain-rule vector $\nabla_{\tilde{\theta}} \mathcal{L}_{\text{meta}}$ can be held constant while forming the vector–Jacobian product; the dependence of $\tilde{\theta}$ on ϕ through the weights must remain. This local approximation avoids unrolling a long inner trajectory, but does not provide an error bound relative to the fully optimized bilevel objective.

A.4 Grid coverage of the strategy space

A pairing strategy is parameterized by $(c, r) \in \mathbb{R}^2_{>0}$, selecting responses near $\mu_x + c\sigma_x$ and $\mu_x - r\sigma_x$. Within $[1, 3] \times [1, 3]$, every target location lies within ℓ_∞ distance $1/2$ of the grid $\{1, 2, 3\}^2$. This describes coverage of target positions only; it does not bound the error introduced by selecting finite sampled responses or guarantee that the best pairing strategy lies in this range. Empirically, refining to 5×5 or 7×7 yields only marginal gains ($+0.1$, $+0.1$ on AlpacaEval 2.0; Appendix D.10), supporting the practical choice of the coarser grid.

A.5 Optimization landscape and convergence

We do not claim formal convergence guarantees: the inner objective is non-convex (DPO on a transformer LLM), and the sampled data change with the policy. The compact meta-network reduces the dimension of the meta-optimization problem, and the one-step update avoids differentiating through a long optimization trajectory. Neither property guarantees a well-conditioned objective or bounded gradients. The ReLU network is piecewise differentiable. Empirically, the meta-loss decreases overall (Fig. 3a), and the main AlpacaEval results report standard deviations no greater than 0.5 percentage points across seeds.

A.6 Discussion on distribution assumptions

The grid construction uses μ_x, σ_x to summarize the location and scale of the response-reward distribution. These statistics do not fully characterize arbitrary reward distributions. To examine their empirical shape, we analyzed reward distributions on Anthropic-Helpful.

Table 4: Distribution of reward-distribution types in Anthropic-Helpful (sampled with Llama-2-7B-Chat, $n = 200$ per prompt).

Distribution Type	Proportion	Identification Criterion
Unimodal	91.7%	Kurtosis < 4, single mode
Bimodal	5.8%	Two distinct modes
Heavy-tailed	2.5%	Kurtosis > 6

The vast majority (91.7%) of prompts exhibit unimodal distributions, although unimodality alone does not imply that μ_x, σ_x fully describe them. For multimodal or heavy-tailed prompts, META-PAP's learned weighting downweights pairs that fall in sparse regions. Excluding non-unimodal prompts entirely yields only $+0.4$ on AlpacaEval 2.0 (Llama-2-7B), so the method is robust to assumption violations.

B Architecture and Implementation Details

B.1 Complete hyperparameter configuration

Table 5: Complete hyperparameter settings. Identical configurations across models except for the policy learning rate.

Hyperparameter	Llama-3-8B	Mistral-7B-v0.3	Llama-2-7B
Policy optimization			
Learning rate α	1×10^{-6}	1×10^{-6}	5×10^{-7}
Optimizer	AdamW	AdamW	AdamW
Weight decay	0.01	0.01	0.01
LR scheduler	Cosine	Cosine	Cosine
Warmup steps	100	100	100
Gradient clipping	1.0	1.0	1.0
DPO temperature β	0.1	0.1	0.1
Batch size	64	64	64
Meta-network training			
Learning rate η	1×10^{-4}	1×10^{-4}	1×10^{-4}
Optimizer	AdamW ($\beta_1=0.9, \beta_2=0.999$)	—	—
Weight decay	0.0	0.0	0.0
Architecture	MLP $5\to64\to32\to1$ with ReLU and final sigmoid		
Parameters	2,497		
Meta batch size	64	64	64
Data and generation			
Samples per prompt n	200	200	200
Sampling temperature	0.8	0.8	0.8
Top-p	0.95	0.95	0.95
Max response length	512	512	512
Grid size	3×3	3×3	3×3
Avg. pairs per prompt	$\sim$8	$\sim$8	$\sim$8
Training schedule and meta-set			
Epochs	1	1	1
Meta-set size M	150	150	150
Reward-gap threshold	0.3	0.3	0.3
Precision	bfloat16	bfloat16	bfloat16

B.2 Meta-network architecture

The meta-network V_ϕ is intentionally compact to avoid overfitting to the small meta-set and to keep the meta-gradient computation cheap. Forward pass:

$$\mathbf{h}_1 = \mathrm{ReLU}(\mathbf{W}_1\mathbf{f} + \mathbf{b}_1) \in \mathbb{R}^{64}, \quad \mathbf{h}_2 = \mathrm{ReLU}(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2) \in \mathbb{R}^{32}, \quad w = \sigma(\mathbf{w}_3^\top \mathbf{h}_2 + b_3). \tag{9}$$

Total parameters: $5\cdot64 + 64 + 64\cdot32 + 32 + 32 + 1 = 2{,}497$. Initialization: Xavier uniform for weights, zero biases. The output bias b_3 is initialized to 0, with outputs intended to start near 0.5. This is not an exact equal-weight initialization: the nonzero Xavier-initialized weights can produce input-dependent outputs, and constant weights of 0.5 would scale the objective relative to the $w \equiv 1$ Uniform Grid baseline.

B.3 Computational infrastructure

All experiments use a single node with 8×NVIDIA A100 80GB GPUs and 512GB CPU RAM. The implementation uses PyTorch 2.1, Transformers 4.36, and DeepSpeed ZeRO-3 for memory partitioning. Approximate per-epoch wall-clock times: Llama-3-8B 18.5h, Mistral-7B 16.8h, Llama-2-7B 16.2h. Vanilla DPO on Llama-2-7B uses 12.0h per epoch, so the overhead of the full META-PAP pipeline is $\sim$35%. As discussed in Sec. 4.5 and Appendix D.9, this overhead is dominated by on-policy generation and reduces to $\sim$15% at $n=32$.

B.4 Engineering considerations

Numerical stability. Reward differences in the meta-set can be small for some prompts. We add a small constant $\epsilon = 10^{-3}$ to σ_x in feature normalization (Eq. 4, second and third components) to avoid division-by-zero on prompts with extremely concentrated distributions. We also clip the meta-gradient $\nabla_\phi \mathcal{L}_{\text{meta}}$ by global norm 1.0, matching the gradient-clipping setting on the policy.

Memory. The meta-network has few parameters, but this alone does not determine the memory required for the virtual policy update and meta-loss evaluation. We stream-generate responses prompt-by-prompt and re-use the same pool for all K grid pairs of that prompt.

Reproducibility. We fix random seeds for response sampling, grid construction, and DataLoader shuffling. The meta-network's w outputs are deterministic given the same $(\theta, \phi, \mathbf{F})$. Across three seeds, the standard deviation of the win rate is $\leq 0.5\%$ on AlpacaEval 2.0.

C Data, Metrics, and Baselines

C.1 Data splits and reward-model details

Training prompts. Anthropic-Helpful provides 43,835 preference pairs; we use only the prompts (discarding the labels) since META-PAP generates its own pairs from on-policy samples. UltraFeedback [8] has 61,135 pairs; we likewise use only its prompts.

Reward model. The default reward model is **ArmoRM-Llama3-8B-v0.1** [31], a public multi-objective reward model used to score all n on-policy responses. These scores are used for meta-set construction and per-pair feature extraction. We do not use oracle/GPT-4 signals for policy training or meta-set construction; Appendix D.8 compares high-confidence filtering with human screening, and Appendix D.6 verifies that the gains transfer across reward-model choices (UltraRM-13B, Starling-RM-7B).

Meta-set construction. The meta-set is disjoint from training prompts and from all evaluation benchmarks. We retain pairs with $\Delta r > 0.3$ (retains $\sim$60% of candidates) and stratify by $\sigma_x \in \{<1.5, [1.5, 3.0], >3.0\}$, sampling 50 pairs per stratum to obtain $M = 150$ balanced meta-pairs.

C.2 Metric definitions

AlpacaEval 2.0 [9] reports the length-controlled win rate against GPT-4-Turbo on 805 instruction-following prompts. **Arena-Hard** [18] measures the win rate on 500 adversarially-curated prompts that elicit reasoning and difficult instruction-following. **MT-Bench** [36] evaluates multi-turn conversation across 80 prompts in 8 categories on a 1–10 scale judged by GPT-4. **WildBench** [20] provides 1,024 real-world user queries scored by GPT-4-Turbo. The main-results table reports mean $\pm$ standard deviation over three seeds; supplementary tables without uncertainty entries report point estimates.

C.3 Baseline configurations

Vanilla DPO: max-min pair from $n = 200$ responses per prompt, matching the per-prompt sample count used by META-PAP. The offline baseline and on-policy methods differ in data refresh (Appendix D.7). **Static-$\mu \pm 2\sigma$**: select responses closest to $\mu_x + 2\sigma_x$ and $\mu_x - 2\sigma_x$. **Reward-Gap Weighting**: weight each pair by $\sigma(\Delta r/\tau)$ with $\tau = 1.0$. **Top-50% Filtering**: keep pairs with Δr above the median. **Curriculum-by-Gap**: linear schedule from small-gap to large-gap pairs over the epoch. **RSO** [21]: statistical rejection sampling with the official hyperparameters. **Online DPO** [13]: pairs regenerated against the current policy at each step using the same reward model as META-PAP. **Iterative DPO**: three rounds of (sample $\rightarrow$ label by reward model $\rightarrow$ DPO update), with each round initializing from the previous policy. **SPIN** [7]: self-play fine-tuning following the public reference implementation. **Uniform Grid**: same 3×3 grid as META-PAP but with $w \equiv 1$. **Oracle**: train nine separate DPO models, one for each grid cell, using that cell's pairs across all training prompts. For each prompt in the held-out meta-set, the diagnostic allocation selects the cell-specific model with the lowest DPO loss. Cell selection therefore uses meta-set loss rather than downstream benchmark

scores; the meta-set is disjoint from the evaluation benchmarks. The nine-model cost refers to nine training runs in total. Oracle serves as a diagnostic grid-cell reference, not a formal upper bound on learned mixtures.

C.4 Mean and standard-deviation reporting

The main-results table reports mean $\pm$ standard deviation across three random seeds. Supplementary tables without uncertainty entries report point estimates. Seeds control: response sampling, meta-set sub-sampling within strata, DataLoader shuffling, and meta-network initialization. Standard deviations on AlpacaEval 2.0 are $\leq 0.5\%$ for all methods on all models, indicating stable training under the bilevel framework.

C.5 Motivation experiment details

The motivation experiment in Section 3.1 (Table 1) is conducted on Llama-2-7B-Chat with $n = 200$ on-policy responses per prompt and AlpacaEval 2.0 length-controlled win rate as the metric. We use Llama-2-7B-Chat (instruction-tuned) for the motivation experiment because it generates more coherent on-policy responses for reliable reward scoring; the main experiments use base models with SFT, and we verified that the same ordering of gap rules within each stratum holds on the SFT base model. Figure 2a reports gains over Vanilla DPO for the SFT-initialized Llama-2-7B model used in the main experiments, whereas Table 7 reports absolute win rates from the separate Llama-2-7B-Chat motivation experiment.

Bucket counts. The diversity strata are populated as follows on the 11,010 Anthropic-Helpful prompts:

Table 6: Prompt counts per response-diversity stratum.

Stratum	Threshold	Count	Fraction
Low-diversity	$\sigma_x < 1.5$	4,218	38.3%
Medium-diversity	$1.5 \leq \sigma_x \leq 3.0$	4,891	44.4%
High-diversity	$\sigma_x > 3.0$	1,901	17.3%

Per-cell numbers. Table 1 reports only the best/worst gap and their difference. The full 3×3 table of (stratum $\times$ gap rule) is:

Table 7: Full per-cell AlpacaEval 2.0 length-controlled win rate (%) on Llama-2-7B-Chat. Best per row is bold. The relationship inverts from low to high diversity.

Stratum	Small gap	Medium gap	Large gap
Low ($\sigma_x < 1.5$)	11.4±0.4	19.5±0.4	**23.7±0.4**
Medium (1.5–3.0)	14.6±0.4	**17.1±0.4**	10.3±0.5
High ($\sigma_x > 3.0$)	**16.8±0.5**	13.4±0.5	8.1±0.6

Threshold sensitivity. The choice of σ_x thresholds (1.5, 3.0) was made before observing the per-cell numbers. To verify the cross-over pattern is not an artifact of these thresholds, we re-ran the experiment under two alternative threshold sets:

Table 8: The cross-over (large-gap best on low-diversity, small-gap best on high-diversity) is robust to threshold choice.

Thresholds (low / high)	Best on low	Best on high	Pattern holds?
$\sigma_x < 1.0$ / $\sigma_x > 2.5$	Large	Small	✓
$\sigma_x < 1.5$ / $\sigma_x > 3.0$ (default)	Large	Small	✓
$\sigma_x < 2.0$ / $\sigma_x > 3.5$	Large	Small	✓

D Additional Experimental Results

D.1 Cross-domain generalization

Table 9: Cross-domain generalization on Llama-2-7B (AlpacaEval 2.0 win rate). META-PAP's improvements transfer with only modest degradation, indicating the learned weighting captures distributional structure rather than memorizing specific prompt-response patterns.

Train domain	Test domain	Static-$\mu\pm2\sigma$	META-PAP
Helpful	Helpful (in-domain)	14.6	17.1 (+2.5)
	Harmless (out-of-domain)	16.3	18.7 (+2.4)
	UltraFeedback (out-of-domain)	18.4	20.9 (+2.5)
UltraFeedback	UltraFeedback (in-domain)	19.2	22.1 (+2.9)
	Helpful (out-of-domain)	14.1	16.4 (+2.3)
	Harmless (out-of-domain)	15.8	18.0 (+2.2)

D.2 Noise tolerance

Table 10: Robustness to noisy meta-set labels (Llama-2-7B). META-PAP tolerates up to 20% label noise while still beating the static baseline; at 50% label noise, performance approaches the static baseline.

Noise level	AlpacaEval 2.0	Arena-Hard	MT-Bench
0% (clean)	17.1	11.9	7.08
10% flips	16.7	11.5	6.98
20% flips	16.2	11.0	6.87
30% flips	15.6	10.4	6.74
50% flips (random)	14.8	9.6	6.58
Static baseline (clean)	14.6	9.4	6.58

D.3 Response-distribution statistics

Table 11: Response-distribution statistics by diversity stratum. Low-diversity prompts have near-zero skewness and Gaussian-like kurtosis; high-diversity prompts are slightly right-skewed and heavier-tailed, but still predominantly unimodal.

Statistic	Low σ	Medium σ	High σ
Mean reward μ	2.31	1.84	0.97
Std reward σ	0.89	2.18	3.92
Skewness	-0.08	0.24	0.61
Kurtosis	2.84	3.12	3.87
% Unimodal	96.2%	92.4%	86.8%

D.4 Extended training

Table 12: Extended training (Llama-2-7B, AlpacaEval 2.0). META-PAP's advantage persists and slightly grows with longer training, confirming it learns a fundamentally better weighting policy rather than only accelerating convergence.

Method	1 Epoch	2 Epochs	3 Epochs
Vanilla DPO	13.2	13.9	14.3
Static-$\mu\pm2\sigma$	14.6	15.4	15.9
META-PAP	17.1	18.2	18.8
Δ (vs. Static)	+2.5	+2.8	+2.9

D.5 Scaling analysis

Table 13: Scaling analysis (AlpacaEval 2.0). †70B uses LoRA (rank 64) due to memory. Gains shrink slightly with model size as larger models become more robust to the underlying pairing rule, but remain substantial.

Model size	Vanilla DPO	META-PAP	Δ
Llama-2-7B	13.2	17.1	+3.9
Llama-2-13B	16.8	20.5	+3.7
Llama-2-70B†	24.3	27.6	+3.3

D.6 Reward-model sensitivity

The default reward model is ArmoRM-Llama3-8B-v0.1. To verify that META-PAP's gains do not depend on a specific reward-model choice, we re-trained Llama-2-7B with two alternative public reward models—UltraRM-13B [8] and Starling-RM-7B [38]—using identical training pipelines.

Table 14: Reward-model sensitivity on Llama-2-7B (AlpacaEval 2.0 win rate). In each run, META-PAP uses the same reward model for candidate scoring and meta-set construction.

Reward model	Static-$\mu\pm2\sigma$	META-PAP	Δ over static
ArmoRM-Llama3-8B (default)	14.6	17.1	+2.5
UltraRM-13B	14.3	16.8	+2.5
Starling-RM-7B	14.1	16.5	+2.4

Across the three reward models, META-PAP improves AlpacaEval 2.0 win rates by 2.4–2.5 points over the static baseline.

Cross-reward-model meta-supervision. To separate training-side scoring from meta-supervision, we keep ArmoRM fixed for candidate scoring and feature extraction and construct the meta-set with a different reward model. On Llama-2-7B, the cross-model results remain close to the default setting (Table 15). This supports robustness to the source of meta-supervision, but does not establish immunity to biases shared by multiple reward models.

Table 15: Cross-reward-model meta-supervision on Llama-2-7B. Training-side scoring uses ArmoRM throughout. Entries are point estimates without uncertainty estimates. A dash denotes no meta-set for Static and an unreported Arena-Hard score for Starling.

Method	Meta-set reward model	AlpacaEval 2.0	Arena-Hard
Static-$\mu\pm2\sigma$	—	14.6	9.4
META-PAP	ArmoRM (default)	17.1	11.9
META-PAP	UltraRM-13B	17.0	11.7
META-PAP	Starling-RM-7B	16.8	—

D.7 Comparison with on-policy and iterative DPO baselines

Because META-PAP samples from the current policy at training time, the most natural reference class is on-policy and iterative DPO methods. We compare against three such baselines on Llama-2-7B with identical SFT initialization, reward model, and evaluation protocol:

Table 16: Comparison with on-policy and iterative DPO methods on Llama-2-7B. META-PAP outperforms all three. The two axes (data freshness vs. within-batch pair quality) are orthogonal and could be combined.

Method	AlpacaEval 2.0	Arena-Hard	Notes
Vanilla DPO	13.2±0.3	8.7±0.2	offline, max-min
Online DPO [13]	15.4±0.4	10.1±0.3	on-policy, single iteration
Iterative DPO (3 iterations)	15.8±0.4	10.4±0.3	sample→label→DPO × 3
SPIN [7]	14.9±0.4	9.8±0.3	self-play
META-PAP	**17.1±0.4**	**11.9±0.3**	on-policy, learned weighting

Interpretation. Online and Iterative DPO improve over Vanilla DPO by mitigating the SFT-vs-policy distribution shift—they refresh the data so the policy trains against pairs sampled from a distribution close to its current state. META-PAP addresses a different issue: even when the data is fresh, not every sampled pair carries equal signal, and a single max-min rule wastes most of the on-policy responses. The two mechanisms therefore compose naturally; an Iterative+META-PAP variant would refresh the pool every T steps and re-weight pairs within each pool, which we leave to future work.

D.8 Meta-dataset construction comparison

Table 17: Meta-dataset construction comparison (Llama-2-7B). High-confidence filtering and human screening with 150 retained pairs give similar point estimates (17.1 and 17.3). Uncertainty estimates are not reported for these comparisons.

Construction method	AlpacaEval 2.0	Human-screened pairs used
High-confidence filtering (default)	17.1	—
Random sampling	15.7	—
Bootstrapping (model self-judgment)	16.4	—
Human screening	16.8	50
Human screening	17.3	150

Human-screening protocol. Three NLP graduate students who were not authors each independently screened the same 200 candidate pairs from the stratified pool, yielding 600 individual judgments. They marked each pair as showing a "clear quality difference" or as "ambiguous." A pair was eligible if at least two annotators marked it as clear. In total, 153 pairs met this criterion. The two human-screening conditions in Table 17 used 50 and 150 pairs from this pool, respectively.

D.9 Computational breakdown

Table 18: Per-epoch computational breakdown (Anthropic-Helpful, Llama-2-7B, 8×A100). Component rows describe META-PAP at $n = 200$; the 720-min row is the separately reported offline Vanilla DPO baseline. Runtime ratios use this fixed Vanilla DPO reference configuration, including the reported 15% overhead at $n = 32$. That comparison changes only the sampling count of META-PAP.

Component	Time (min)	Percentage
Response generation (200/prompt)	692	71.2%
Reward-model scoring	94	9.7%
Grid pair construction	38	3.9%
Feature extraction	12	1.2%
Meta-network forward/backward	18	1.9%
DPO loss computation	86	8.9%
Optimizer steps	32	3.3%
Total (META-PAP)	972	100%
Vanilla DPO baseline	720	—
Overhead	+252	+35%

D.10 Hyperparameter sensitivity

Table 19: Hyperparameter sensitivity (Llama-2-7B, AlpacaEval 2.0). META-PAP is robust within $\pm 25\%$ of each default; refining the grid beyond 3×3 yields negligible gains, justifying the default choice.

Hyperparameter	-50%	-25%	default	$+25\%$
Meta-LR η	16.2	16.8	**17.1**	16.9
Reward-gap threshold	16.6	16.9	**17.1**	17.0
Stratum size	16.5	16.9	**17.1**	17.1
Inner LR α	16.4	16.8	**17.1**	16.7
Grid resolution: 1×1 / 3×3 / 5×5 / 7×7	15.4 / **17.1** / 17.2 / 17.2			

D.11 Comparisons using a shared candidate grid

Table 20 compares selection and weighting rules on Llama-2-7B using the same approximately eight candidates per prompt from the 3×3 grid. Best-of-grid and random selection retain one pair, whereas the weighting methods train on all retained grid pairs. Thus the candidate pool is controlled, but the single-pair rows do not match the number of optimized pairs or total compute. Among the multi-pair methods, learned weighting improves over both uniform and reward-gap weights.

Table 20: Shared-candidate-grid comparison on Llama-2-7B. Entries are supplementary point estimates.

Method	Pairs used per prompt	AlpacaEval 2.0	Arena-Hard
Best-of-grid (largest Δr)	1 selected	14.9	9.8
Random-from-grid	1 selected	14.1	9.2
Reward-Gap Weighting on grid	~ 8, weighted	15.3	10.2
Uniform Grid	~ 8, uniform	15.6	10.4
META-PAP	~ 8, learned	17.1	11.9

D.12 Weight normalization

The default sigmoid weights are not normalized, so they can change both the relative contribution of pairs and the overall gradient scale. Table 3 in Sec. 4.7 compares three normalized variants on Llama-2-7B. Their gains over Uniform Grid persist, suggesting a contribution from relative pair weighting beyond global rescaling. These comparisons do not establish a unique causal decomposition of selection and scale effects.

E Qualitative Analysis

E.1 Case studies

Table 21: Case studies of META-PAP's prompt-aware weighting. The pattern matches Sec. 4.6: large-gap pairs get high weights when σ_x is small (clear quality contrast); high-σ_x extreme pairs are softly rejected (rejected response is an outlier), and moderate-gap pairs near the bulk are favored.

Prompt (diversity)	Chosen response (excerpt)	Rejected response (excerpt)	Weight
Low-diversity example ($\sigma_x = 0.82$): "What is the capital of France?"			
Large-gap pair	"The capital of France is Paris, a major European city and global center for art, fashion…"	"I'm not entirely sure, but I think it might be Lyon or possibly Marseille…"	**0.87**
Small-gap pair	"Paris is the capital of France."	"The capital of France is Paris, located in the north."	0.31
High-diversity example ($\sigma_x = 3.41$): "What are the ethical implications of AI in healthcare?"			
Large-gap pair	"AI in healthcare raises profound ethical questions spanning privacy, bias, accountability…" [400 words]	"AI bad for healthcare because robots."	0.24
Small-gap pair	"Key ethical concerns include patient data privacy and algorithmic fairness…" [balanced, 180 words]	"Important considerations are consent, transparency, and equitable access…" [similar quality, 160 words]	**0.79**

E.2 Feature importance via gradients

Table 22: Feature importance measured by average gradient magnitude during training. The ranking matches the ablation order in Fig. 2b, confirming that the meta-network's behavior is indeed driven by the reward gap and prompt-level diversity.

Feature	Avg. $\lvert\partial w/\partial f_i\rvert$	Rank
Reward gap $r_w - r_l$	0.312	1
Response diversity σ_x	0.247	2
Chosen position $(r_w - \mu_x)/(\sigma_x + \epsilon)$	0.168	3
Rejected position $(r_l - \mu_x)/(\sigma_x + \epsilon)$	0.142	4
Training progress t/T_{total}	0.098	5

Higher-order distribution features. A natural question is whether the 5-dimensional feature set captures everything useful, or whether higher-order distribution shape features (skewness, kurtosis, percentile ranks) would help. We tested two extensions on Llama-2-7B:

- + **skewness** + **kurtosis** (7-D feature): 17.3 on AlpacaEval 2.0 vs. 17.1 for the default 5-D feature, a +0.2 improvement that is within seed noise.
- + **length, perplexity, log-prob ratio** (8-D feature): 17.0, no improvement and increased variance across seeds.

We retain the 5-D design because the small meta-set ($M = 150$) makes higher-dimensional features prone to overfitting, and the +0.2 skewness/kurtosis gain does not justify the increased complexity. This is consistent with the design principle in Sec. 3.6: the relevant signal lies in the *interaction* between Δr and σ_x, which the 5-D MLP can already learn.

E.3 Failure cases

Extremely uniform distributions ($\sigma_x < 0.3$, $\sim$2% of prompts). When all responses have nearly identical rewards, no pair carries useful contrast and META-PAP defaults to near-uniform weights, matching Vanilla DPO.

Multimodal distributions ($\sim$6% of prompts). Bimodal response distributions (e.g. "Should I invest in stocks or bonds?" eliciting both pro-stock and pro-bond clusters) make μ_x, σ_x uninformative. Grid

positions can fall in sparse intermediate regions; META-PAP partially mitigates this via low weights, but performance is slightly reduced.

Severe domain shift. If the meta-set differs substantially from the target distribution, the learned weighting may transfer less effectively. The reported Helpful and UltraFeedback transfer directions in Appendix D.1 show different degrees of degradation; they do not establish robustness to arbitrary domain shift.

E.4 Isolated gap rules and learned mixtures

Table 1 and Figure 5b examine different training settings. The table compares policies trained exclusively on one of three gap rules, whereas the heatmap summarizes weights learned when all nine grid cells are available simultaneously. These settings need not favor the same gap size.

Isolated gap rules. On high-diversity prompts, the small-gap rule gives the highest win rate among the three separately trained policies in Table 1, and the large-gap rule gives the lowest. This comparison measures the performance of each rule when it supplies all training pairs.

Learned mixtures. Figure 5b assigns the largest high-diversity weight to medium-gap pairs within a weighted mixture. Such a weight reflects a pair's contribution in the presence of other grid pairs; it is not the performance of a policy trained only on that gap. In the reported high-diversity comparison, the heatmap mixture improves over training only on small-gap pairs by $+1.4$ points.

Both analyses favor non-extreme contrasts on high-diversity prompts. The difference between the best isolated rule and the largest mixture weight reflects the different comparison settings. Reduced sensitivity to tail outliers is a possible explanation, but these comparisons do not directly measure reward-model error for individual tail responses.

F Sample-Efficiency Analysis

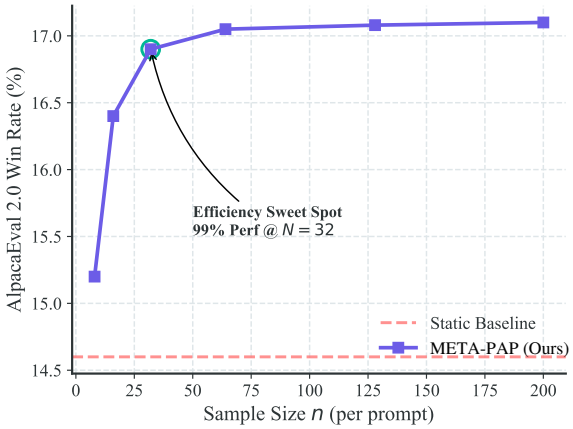


Figure 7: **Sample-efficiency analysis** on Llama-2-7B. Performance saturates rapidly as n increases. Using only $n = 32$ samples retains $\approx 99\%$ of the win rate of the full $n = 200$ setting, dramatically reducing the dominant cost in Table 18.

Sampling n responses per prompt is the main computational cost of distribution-aware pair construction. Figure 7 shows rapid convergence: increasing n from 4 to 32 yields most of the gain, and the marginal benefit beyond $n = 50$ is negligible. Practically, μ_x, σ_x are reliable distribution statistics with relatively few samples, so practitioners can deploy META-PAP with $n \in [32, 50]$ with a small change in performance. At $n = 32$, the reported overhead is 15% relative to the fixed Vanilla DPO reference configuration, compared with 35% at $n = 200$.

G Extended Related Work

On-policy vs. offline DPO. Online DPO variants [13] regenerate preference pairs against the current policy to mitigate the distribution shift between the SFT-time pairs and the policy being optimized. META-PAP's grid generation is on-policy in this sense, but its central contribution is orthogonal: even with on-policy pairs, the question of *which* on-policy pair is most informative remains, and is what META-PAP addresses through prompt-aware weighting.

Reward-model overoptimization. Gao et al. [12] study overoptimization against a proxy reward model. The diversity-stratified pattern in Sec. 3.1 motivates examining the reliability of extreme pairs, but our reward-distribution statistics do not independently verify reward-model correctness.

Bilevel optimization for LLMs. Gradient-based data selection has been studied for instruction tuning [33], while validation-guided meta-reweighting originates in supervised learning [26, 28]. META-PAP extends this line by applying bilevel optimization to preference pair selection in DPO with a small, structurally informed feature space and a fully automated meta-set, making it deployable at scale.

Gradient-based and token-level weighting. DoGE [11] uses a bilevel generalization signal to learn domain sampling weights. This shares the gradient-alignment motivation of Appendix A.2, but operates at the domain level, whereas META-PAP learns a feature-conditioned function over preference pairs. Selective Preference Optimization (SePO) [34] estimates token-level rewards to select informative tokens. Pair weighting and token selection operate at different granularities and could be combined; their relative merits depend on the selection and training protocol.

Process and step-level supervision. For reasoning tasks, process-reward models [19, 32] provide step-level signals beyond outcome rewards. Although META-PAP targets the response level, the same prompt-conditional weighting question arises at finer granularity, suggesting an extension to process-level pair selection.

H Limitations and Broader Impact

Limitations. META-PAP has three main limitations. First, it relies on a proxy reward model both for response scoring and for meta-set construction; if the reward model has systematic bias (e.g., length preference), the learned weighting can inherit it. Appendix D.8 also evaluates human screening of candidate meta-pairs. Cross-reward-model supervision (Appendix D.6) tests sensitivity to the meta-label source but does not rule out shared biases or adversarial reward-model errors. Second, the fixed grid summarizes location and scale; most measured distributions are unimodal (Table 4), but this does not guarantee adequate coverage, especially in multimodal cases. Third, on-policy sampling adds compute overhead; Sec. 4.5 shows this can be reduced to $\sim$15% at $n = 32$, but for very large models the absolute cost remains non-trivial.

Broader impact. META-PAP is a training-time optimization method for preference-based alignment. It does not change the alignment target (which is set by the reward model and the meta-set), only the efficiency with which the policy moves toward that target. Two implications are worth noting. (i) By making preference optimization more sample-efficient, META-PAP may reduce the human-annotation burden for safety-critical alignment use cases. (ii) The same machinery could be used to amplify a biased reward signal more effectively; we believe this risk is mitigated by the meta-set being open to human inspection (only 150 pairs) and by the interpretable, low-dimensional feature space, which makes auditing the learned weighting policy straightforward.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Introduction state the empirical, methodological, and experimental claims; Sections 3–4 and Appendices A, D provide the corresponding method, experiments, and analysis.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations and broader impacts are discussed in Appendix H, covering reward-model bias inheritance, limitations of the response-distribution summaries, and the on-policy generation overhead.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Appendix A states the local assumptions and derives the one-step gradient-alignment identity.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 and Algorithm 1 describe the pair construction, weighting procedure, and bilevel updates. Section 4 and Appendices B–C specify the training settings, baselines, and evaluation metrics used in the main experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce

the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The META-PAP source code is not yet public, and we plan to release it upon publication. The datasets and base models are publicly available, and Section 3, Algorithm 1, and Appendices B–C describe how preference pairs are constructed, how the policy and meta-network are trained, and how the models are evaluated.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and Appendix B specify training data, test benchmarks, on-policy sampling settings, hyperparameters, optimizer choices, meta-set construction, and seed count.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 2 reports mean ± standard deviation over three random seeds, and Appendix C.4 describes the sources of randomness.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix B reports the hardware, software stack, and per-epoch wall-clock times, and Table 18 provides a component breakdown. A total compute budget for the complete experimental campaign is not reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The work uses public datasets and open model families to study sample efficiency in preference-based LLM alignment, and we discuss risks, safeguards, and limitations in Appendix H.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix H discusses positive impacts (reduced annotation burden through automated meta-set construction) and negative dual-use risks (the same machinery can amplify a biased reward signal), along with mitigations through low-dimensional auditable features and an inspectable meta-set.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new high-risk pretrained model or scraped dataset.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The original creators of the datasets, model families, and methods are credited through citations. A complete asset-by-asset account of licenses and terms of use is not provided in the manuscript.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new dataset, trained model, or implementation package is released with this manuscript.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [No]

    Justification: Appendix D.8 describes the annotator background, pair-screening task, and majority-agreement criterion. The paper does not reproduce the full participant instructions or report compensation details.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [No]

    Justification: The manuscript reports a human-annotation comparison but does not specify an institutional review or exemption determination, participant-risk disclosure, or consent procedures.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Section 4 and Appendices B–C describe the LLM policy models, reward models used for scoring and meta-set construction, and LLM-based benchmark evaluation.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.