Truncate Bad, Upweight Good: BoN-Style Distillation via Rank-Based Classification

Yarin Bar[1] **Yaniv Romano**[1,2]

[1]Department of Electrical and Computer Engineering

[2]Department of Computer Science

Technion–Israel Institute of Technology

yarinbar@campus.technion.ac.il yromano@technion.ac.il

Abstract

Inference-time selection methods, such as Best-of-N, improve generation by sampling a pool of candidates and selecting the top-ranked completion according to a reward model. Distillation seeks to amortize this procedure into a single policy by replacing raw rewards with in-pool ranks and learning a policy that upweights higher-ranked completions. However, existing rank-based policies typically use smooth full-support reweighting, so low-ranked completions receive less mass but remain in the target support. Although a sharper reweighting reduces lower-tail mass, it also increases reliance on brittle ranking at the top made by a single reward model. We propose **TUP**: a **T**runcate-bad, **U**pweight-good **P**olicy that removes low-ranked completions from the support and reweights only the retained upper tail with a tunable sharpness. TUP admits a closed-form, prompt-independent normalization and can be trained fully offline via binary cross-entropy, using shifted-truncated win-rates as soft labels and distilled-to-reference log-likelihood ratios as logits. Theoretically, under certain assumptions, we show that for any unknown oracle reward, the best monotone rank-reweighting can be matched by a lower-tail truncation rule, providing formal support for removing the lower tail rather than merely downweighting it. Empirically, we show that TUP is competitive with strong offline alignment baselines.

1 Introduction

Inference-time selection methods such as best-of-N (BoN) have emerged as an effective tool for improving language models' generation quality [4, 15]. BoN samples several completions from a base policy, scores them using a reward model, and returns the highest-scoring one [1, 17, 36]. The price is computational, since each prompt requires multiple generated completions and reward evaluations [5]. This has motivated a growing line of work on BoN-style distillation [3, 4, 8, 14, 26, 34], where rather than carrying out selection during deployment, one trains a single policy to internalize the behavior of the inference-time selector.

A key insight from recent BoN-style distillation work is that the relevant training signal for a completion is not its raw reward in isolation, but its relative rank within the sampled pool. This relative rank is often referred to as the completion *win-rate*. In turn, existing distillation methods aim to maximize the win-rate while penalizing the KL distance from the base policy [3, 14, 26]. This results in policies that apply smooth full-support reweighting, favoring higher-ranked completions while keeping every completion in the target support. Sharper reweighting reduces the probability mass of clearly low-ranked completions, but it also concentrates more mass on the very top of the reward-model ranking. As a result, over-sharpening can move the policy further from the reference model (smaller KL-penalty), increasing the risk of reward hacking [19, 28, 32, 38]. Since reward

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

models are imperfect proxies for unknown oracle or human preferences [10, 11, 15, 20, 21, 29, 32, 45], recent work shows that coarse preferences tend to persist across different evaluators, while fine-grained distinctions among top-ranked completions are less consistently agreed upon [23, 24, 43].

Consequently, we argue that the design of a BoN-style distillation policy should separate two choices: (i) how much of the lower-ranked tail should be removed from the support, and (ii) how sharply the retained higher-ranked completions should be upweighted. To this end, we propose **TUP**, a **T**runcate-bad, **U**pweight-good **P**olicy for BoN-style distillation. TUP decouples these choices with a shifted-truncated win-rate transform. Completions whose win-rate falls below the threshold are assigned zero mass, while those above it are softly reweighted by their relative ranking. The threshold parameter controls lower-tail truncation, whereas the upper-tail sharpness parameter controls the upweighting strength within the retained set. By introducing separate parameters for lower-tail truncation and upper-tail sharpness, this design preserves the BoN principle of favoring higher-ranked completions while (i) preventing probability mass from being assigned to clearly undesirable completions, and (ii) avoiding over-sharpening due to possible ranking mistakes at the top.

The resulting target policy remains simple to train from offline data. Each prompt is paired with a pool of completions sampled from the reference policy and scored by a reward model. From these scores, we compute empirical in-pool ranks and convert them into shifted-truncated win-rate labels; by subtracting the truncation parameter, we assign the lower-tail completions an exact zero label. Pairing with the logit transformation, we then train the distilled language model with binary cross-entropy (BCE) loss.

Contributions. In this work, we make three key contributions.

1. **A truncate-bad, upweight-good policy for BoN-style distillation.** We propose a rank-based target policy that removes low-ranked completions using a win-rate threshold and softly reweights the retained upper tail.
2. **Theoretical support for lower-tail truncation and within-tail upweighting.** Under certain assumptions, we give two theoretical justifications for TUP, using the oracle win-rate as the performance criterion. First, we show that the best hard lower-tail truncation rule can match the best monotone policy that reweights completions by proxy-reward win-rate. Second, after fixing a truncation threshold, we show that smooth upweighting of retained completions can further improve the oracle win-rate when, informally, higher proxy-reward win-rates tend to correspond to higher oracle win-rates within the retained tail.
3. **Benchmark results and ablations.** We evaluate TUP on the QRPO benchmark training Llama-8B Tülu 3 SFT on UltraFeedbak and Magpie Air and Mistral-7B-Instruct-v0.2 on Magpie Air, comparing to four leading offline-alignment baseline methods. We evaluate performance using three different reward models, both on held-out data and AlpacaEval [9]. Full implementation of TUP and the experiments are available at github.com/yarinbar/truncate-bad-upweight-good.

2 Preliminaries and related works

We denote a language model as a policy $\pi(y \mid x)$ mapping a prompt $x \in \mathcal{X}$ to a distribution over completions $y \in \mathcal{Y}$. Prompts are drawn from a distribution $\mathcal{D}$ over $\mathcal{X}$. Let $\pi_{\text{ref}}(y \mid x)$ denote the reference policy, e.g., obtained via supervised fine-tuning, and let $r(x, y) \in \mathbb{R}$ be a reward function.

The standard post-training alignment objective seeks a policy that maximizes expected reward while remaining close to the reference policy:

$$\pi^{\star}_{r,\beta}(\cdot|x) = \arg\max_{\pi} \mathbb{E}_{y\sim\pi(\cdot|x)}[r(x,y)] - \beta D_{\text{KL}}(\pi(\cdot|x) \,\|\, \pi_{\text{ref}}(\cdot|x)), \tag{1}$$

where $\beta > 0$ controls the reward–KL tradeoff. This KL-regularized reward-maximization objective is a standard formulation in RLHF for language-model alignment [2, 6, 30, 37, 46], and is commonly optimized with online reinforcement-learning methods such as PPO [33] and GRPO [35].

An alternative paradigm, which we follow in this paper, builds directly on the closed-form solution to (1), known as the Gibbs policy:

$$\pi^{\star}_{r,\beta}(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{r(x,y)}{\beta}\right), \quad Z(x) = \mathbb{E}_{Y\sim\pi_{\text{ref}}(\cdot|x)}\left[\exp\left(\frac{r(x,Y)}{\beta}\right)\right], \tag{2}$$

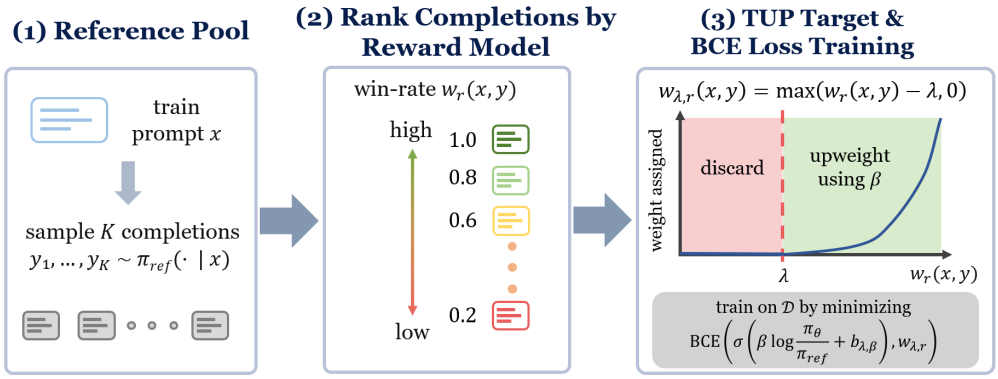

Figure 1: Illustration of TUP.

where $Z(x)$ is the partition function. Rearranging (2) yields

$$r(x, y) = \beta \log Z(x) + \beta \log \frac{\pi^\star_{r,\beta}(y|x)}{\pi_{\text{ref}}(y|x)}. \tag{3}$$

This relation suggests a direct fitting strategy. Replacing the unknown optimal policy $\pi^\star_{r,\beta}$ with a parameterized model π_θ would allow one to train the policy by regressing the right-hand side of (3) to predict the reward. The difficulty is that the partition function $Z(x)$ is prompt-dependent and impractical to compute. Preference-optimization methods avoid the need to compute $Z(x)$ in different but related ways. DPO [31], IPO [13], and SimPO [27] learn from pairwise or binary preference signals, while REBEL [12] regresses relative reward differences between completions.

Although standard post-training alignment typically treats the raw reward $r(x, y)$ as the quantity to optimize, this is not the most natural choice for inference-time selection algorithms such as BoN. Recent inference-aware and rank-based methods, including InfAlign [3] and QRPO [26], motivate replacing raw reward values with relative-rank quantities. A natural choice is the population *win-rate*

$$w_r(x, y) := \mathbb{P}_{Y' \sim \pi_{\text{ref}}(\cdot|x)}\big(r(x, Y') \le r(x, y)\big), \tag{4}$$

which measures how often y beats an independent reference sample. In practice, one can estimate the win-rate from a finite pool of completions.

This win-rate parametrization is useful not only because it matches the relative nature of BoN-style selection, but also because it removes the prompt-dependent partition function in (3). When $Y \sim \pi_{\text{ref}}(\cdot \mid x)$ and the conditional reward distribution is continuous, the probability integral transform implies that $w_r(x, Y) \sim U[0, 1]$. Therefore, for any nonnegative reweighting function $g : [0, 1] \to [0, \infty)$ with finite integral, the rank-based policy

$$\pi_g(y \mid x) = \frac{1}{Z_g} g(w_r(x, y))\pi_{\text{ref}}(y \mid x), \qquad Z_g = \int_0^1 g(u)\, du, \tag{5}$$

has a normalizer Z_g independent of the prompt x. Many rank-based policies follow this construction. For example, classic Best-of-N induces $g_{\text{BoN},N}(w) \propto w^{N-1}$ [4, 14], and the standard QRPO formulation induces $g_{\text{QRPO},\beta}(w) \propto e^{w/\beta}$ [26]. Both assign positive weight over the entire support, leaving open how the win-rate should be transformed into a target-policy reweighting.

This full-support property clarifies how our target differs from related alignment methods. Iterative BoN-distillation methods such as BOND [34] and Faster-WIND [41] also aim to approximate or distill Best-of-N-type behavior, but they inherit the smooth reweighting view rather than defining a hard lower-tail removal rule. Other objectives, including RAFT [8], SLiC-HF [44], and RRHF [42], use reward filtering or relative preference rankings to construct training signals, but they do not specify a prompt-independent rank-transform density that first truncates the lower tail and then softly reweights the retained upper tail. Our work makes this truncation-and-reweighting operation explicit, yielding a finite prompt-independent normalization and a target that can be fit as a soft-classification problem using the transformed win-rate as labels.

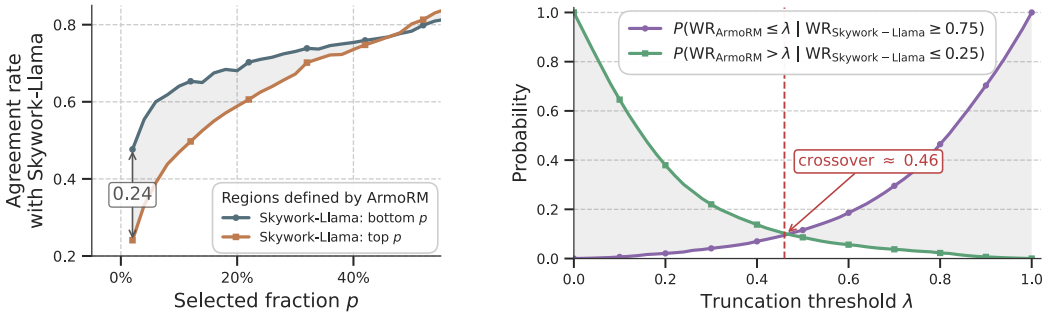

Figure 2: Reward model agreement **(left)** and the cost–benefit tradeoff of λ-truncation **(right)** on UltraFeedback reference completions. We compare ArmoRM, used as the proxy reward model, against the independent Skywork-v2-Llama reward model. **Left:** Fraction of completions placed by ArmoRM in the top or bottom p region that Skywork also places in the same region, as a function of p. **Right:** Probability that λ-truncation discards a Skywork-top-25% completion or retains a Skywork-bottom-25% completion. The crossover marks where these error probabilities are equal.

3 Method

3.1 The proposed truncate-bad, upweight-good target policy

Gibbs-like policies (5) that reweight π_{ref} by a smooth monotone function of the proxy win-rate, such as QRPO [26], face the sharpness tradeoff we discussed above.

Figure 2 (left) provides an empirical illustration motivating our choice to decouple lower-tail truncation from upper-tail sharpness. When the same reference completions are ranked by ArmoRM [39] and by an independent Skywork-v2-Llama reward [25], the models tend to agree more on the bottom of the pool than on the top. Thus, removing the lower tail targets a comparatively stable region across reward models, whereas sharpening toward the top places disproportionate trust in fine-grained top-rank distinctions with weaker cross-model agreement. Additional such comparisons are available in Appendix B.

We instantiate this principle by introducing a truncation level $\lambda \in (0, 1)$, which determines whether a completion remains in the target support, and a sharpness parameter $\beta > 0$, which controls how strongly the retained completions are reweighted. Given λ, we define the *shifted-truncated win-rate*

$$w_{\lambda,r}(x, y) \;:=\; \max\big(w_r(x, y) - \lambda,\, 0\big), \tag{6}$$

and use it to define the transformed reward

$$R_\lambda(x, y) \;:=\; \operatorname{logit}\big(w_{\lambda,r}(x, y)\big) \;=\; \log \frac{w_{\lambda,r}(x, y)}{1 - w_{\lambda,r}(x, y)}. \tag{7}$$

The truncation level λ acts as a quality threshold. Completions with $w_r(x, y) \le \lambda$ are mapped to $R_\lambda(x, y) = -\infty$, while completions above the threshold are retained and scored by the log-odds of their shifted win-rate. Thus, λ imposes a hard quality floor, whereas the transformed reward preserves a smooth ordering among the retained completions. As shown in Figure 2 (right), increasing λ lowers the probability of retaining completions that the independent reward model rank poorly, but also raises the probability of discarding completions this model rank highly.

We now derive the KL-regularized target policy induced by the transformed reward $R_\lambda(x, y)$. The key point is that truncation preserves the tractability of win-rate-based objectives. When the reward distribution is continuous under $Y \sim \pi_{\text{ref}}(\cdot \mid x)$, the population win-rate $w_r(x, Y)$ is uniform on $[0, 1]$. After shifting and truncating this variable, the Gibbs normalizer remains independent of the prompt and can be computed in closed form.

Proposition 3.1 (Closed-form target under the truncated reward)**.** *Assume that, for the fixed prompt x, the distribution of $r(x, Y)$ under $Y \sim \pi_{\text{ref}}(\cdot \mid x)$ is continuous. Then, for every truncation level $\lambda \in (0, 1)$ and regularization parameter $\beta > 0$, the KL-regularized objective in* (1) *with reward R_λ admits the unique Gibbs solution*

$$\pi^\star_{\lambda,\beta}(y \mid x) \;=\; \frac{1}{Z_{\lambda,\beta}}\, \pi_{\text{ref}}(y \mid x) \left(\frac{w_{\lambda,r}(x, y)}{1 - w_{\lambda,r}(x, y)} \right)^{1/\beta}, \tag{8}$$

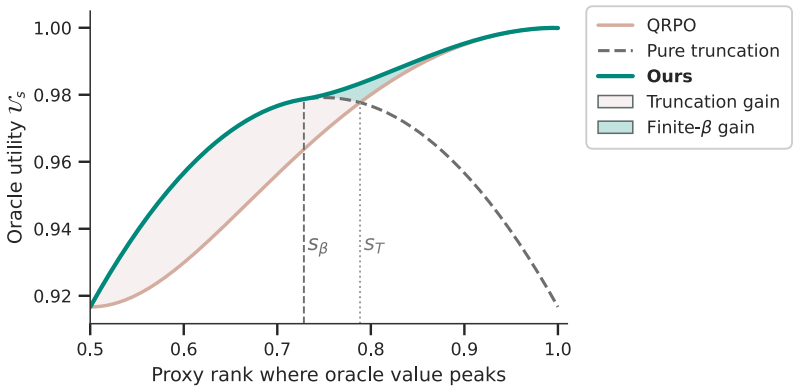

Figure 3: Fixed-threshold illustration for the quadratic un-normalized oracle-proxy profile $m_s(w) = 1 - (w - s)^2$ at $\lambda = 1/2$. The x axis is the proxy-rank location s where oracle utility is maximized, and the y axis shows the unnormalized oracle utility $\mathcal{U}_s$. The curves compare the best QRPO policy, pure truncation, and the best TUP reweighting at the fixed threshold $\sup_{\beta\in(0,\infty]} \mathcal{U}_s(1/2, \beta)$. The vertical dashed line marks s_β, above which finite β improves over pure truncation, and the dotted line marks s_T, where pure truncation no longer outperforms the best QRPO curve.

where the normalization constant is

$$Z_{\lambda,\beta} \;=\; \int_0^{1-\lambda} t^{1/\beta}(1-t)^{-1/\beta}\,dt \;=\; \mathrm{Beta}_{1-\lambda}\big(1+\tfrac{1}{\beta},\,1-\tfrac{1}{\beta}\big). \tag{9}$$

Above, $\mathrm{Beta}_u(a, b)$ *denotes the incomplete Beta function.*

The proposition follows by substituting R_λ into the Gibbs solution in (2); Appendix A.1 gives the population normalizer calculation. The result separates support selection from within-support reweighting, where λ controls which completions remain in the support and the sharpening parameter β controls how strongly the target policy departs from π_{ref} within the retained set.

3.2 Theoretical merits of truncate-bad, upweight-good policies

We now provide a theoretical analysis that sheds light on the roles of truncation and reweighting in our target policy. To set the stage, fix a prompt x, and let $Y \sim \pi_{\mathrm{ref}}(\cdot \mid x)$. Recall that $w_r(x, Y)$ is the win-rate computed using the proxy reward $r(x, \cdot)$. Define also the *oracle win-rate*

$$w_u(x, Y) := \mathbb{P}_{Y'\sim\pi_{\mathrm{ref}}(\cdot|x)}\big(u(x, Y') \le u(x, Y)\big),$$

where $u(x, \cdot)$ is an unknown oracle reward function.

The following analysis builds on the general formulation of rank-based policies π_g from (5). Notably, our proposed policy also falls under this construction, with the function $g_{\lambda,\beta}(w) \propto \big(\frac{w-\lambda}{1-w+\lambda}\big)^{1/\beta}\mathbb{1}\{w > \lambda\}$. With this in place, we can evaluate a candidate policy π_g using the unknown oracle reward, through the expected oracle win-rate of completions Y sampled from π_g:

$$\mathcal{U}_x(g) := \mathbb{E}_{Y\sim\pi_g(\cdot|x)}[w_u(x, Y)].$$

Our first result shows that, among monotone reweightings of proxy rank, it is enough to consider hard truncation rules. Let $\mathcal{F}_\uparrow$ denote the set of nondecreasing densities on $[0, 1]$, and for each $\lambda \in [0, 1)$, let $\nu_\lambda(w) := \frac{1}{1-\lambda}\mathbb{1}\{w \in [\lambda, 1]\}$ denote the uniform density on the upper tail, which is the limit $g_{\lambda,\infty}(w) \propto \mathbb{1}\{w > \lambda\}$. With this notation, we state the following result.

Theorem 3.2 (Informal)**.** *Fix a prompt x, and assume that the proxy reward $r(x, Y)$ is continuous under $Y \sim \pi_{\mathrm{ref}}(\cdot \mid x)$, so that the proxy win-rate $w_r(x, Y)$ is uniform on $[0, 1]$. Let $\mathcal{F}_\uparrow$ be the class of nonnegative nondecreasing densities on $[0, 1]$. Then*

$$\sup_{f\in\mathcal{F}_\uparrow} \mathcal{U}_x(f) = \sup_{\lambda\in[0,1)} \mathcal{U}_x(\nu_\lambda).$$

Algorithm 1 TUP offline BoN distillation with BCE

Require: Offline prompts $\{x_i \sim \mathcal{D}\}_{i=1}^{n}$, reference policy π_{ref}, reward model r, truncation λ, sharpness β

Require: For each x_i, a pool $\{y_{i,j} \sim \pi_{\mathrm{ref}}(\cdot \mid x_i)\}_{j=1}^{K}$ with empirical in-pool win-rates

$$\hat{w}_r(x_i, y_{i,j}) = \frac{1 + \sum_{\ell \neq j} \mathbb{1}[r(x_i, y_{i,j}) \geq r(x_i, y_{i,\ell})]}{K} \in \left\{ \frac{1}{K}, \ldots, 1 \right\}$$

Precompute truncated win-rates and bias term:

$$\hat{w}_{\lambda,r}(x_i, y_{i,j}) = \max\big(\hat{w}_r(x_i, y_{i,j}) - \lambda,\, 0\big), \qquad b_{\lambda,\beta} = \beta \log Z_{\lambda,\beta}$$

for each training step **do**

Sample a minibatch of indexed pool elements $(x_i, y_{i,j})$ from the reference-sampled pool

Compute the logit and predicted probability:

$$s_\theta(x_i, y_{i,j}) = \beta \log \frac{\pi_\theta(y_{i,j} \mid x_i)}{\pi_{\mathrm{ref}}(y_{i,j} \mid x_i)} + b_{\lambda,\beta}, \qquad p_\theta(x_i, y_{i,j}) = \sigma\big(s_\theta(x_i, y_{i,j})\big)$$

Take a gradient step to minimize the BCE loss:

$$\mathcal{L}_{\mathrm{BCE}}(\theta) = -\hat{w}_{\lambda,r}(x_i, y_{i,j}) \log p_\theta(x_i, y_{i,j}) - \big(1 - \hat{w}_{\lambda,r}(x_i, y_{i,j})\big) \log\big(1 - p_\theta(x_i, y_{i,j})\big)$$

end for

The formal statement and proof are given in Appendix A.2.1. This result implies that the best hard-threshold rule, obtained by optimizing λ for each prompt x, can match the best rank-based rule in those families, such as QRPO and BoNBoN. In practice, tuning λ for each prompt x is infeasible, and thus we treat it as a global quality floor. To empirically quantify the potential gain of prompt-adaptive truncation over a fixed global threshold, we consider a simplified, training-free experiment that treats a strong reward model as an "oracle" evaluator. We show that the difference is relatively insignificant. For full details, refer to Section B.3 of the Appendix.

Once the threshold is fixed, we show that tail sharpening can further improve the policy if the proxy win rates within the retained tail is informative about oracle preferences. To formalize this relationship, define the oracle–proxy profile as $m_x(w) := \mathbb{E}_{Y \sim \pi_{\mathrm{ref}}(\cdot|x)}[w_u(x, Y) \mid w_r(x, Y) = w]$, the average oracle win rate among completions with proxy win rate w. In oracle rank space, the following result states that a finite β value can improve over pure truncation when this profile is positively aligned with the within-tail log-odds tilt. The formal statement and proof are given in Appendix A.2.2.

Proposition 3.3 (Informal)**.** *Fix a threshold $\lambda_0 \in (0, 1)$. If, within the retained tail $[\lambda_0, 1]$, the oracle profile m_x has positive covariance with the within-tail log-odds tilt $\log((w - \lambda_0)/(1 - w + \lambda_0))$, then there exists a finite $\beta_0 \in (0, \infty)$ such that $\pi^\star_{\lambda_0,\beta_0}$ achieves strictly larger expected oracle win-rate than pure truncation policy $\pi^\star_{\lambda_0,\infty}$.*

As with the truncation threshold λ, the theoretically optimal tail-sharpening parameter β is prompt-specific. In practice, however, we use a single global value of β across all prompts. In this sense, Theorem 3.2 and Proposition 3.3 provide idealized theoretical support for separating truncation from sharpness, but they do not directly explain how a training model can benefit from it.

To demonstrate the effects of threshold λ and the reweighting effect of a finite β, Appendix A.2.3 and Figure 3 use a stylized, unnormalized quadratic oracle–proxy example; the appendix shows that normalization does not affect the relevant comparisons. The improvement of the pure-truncation policy $\pi^\star_{\lambda,\infty}$ over QRPO reflects the gain from using a threshold to remove low-ranked completions. The improvement of the best finite-β TUP policy over $\pi^\star_{\lambda,\infty}$ reflects the additional gain from reweighting within the retained tail once the threshold is fixed.

Table 1: In-dataset evaluation for Llama 8B Tülu 3 SFT trained separately on UltraFeedback and Magpie Air. We report the LC reward metric for each reward model.

	UltraFeedback eval split			**Magpie Air eval split**		
Method	ArmoRM	Skywork-Llama	Skywork-Qwen	ArmoRM	Skywork-Llama	Skywork-Qwen
Initial	0.1300	9.18±0.24	3.80±0.19	0.1531	10.14±0.30	5.08±0.25
DPO	0.1499	16.16±0.06	7.97±0.03	0.1727	16.33±0.15	9.89±0.05
REBEL	0.1494	16.37±0.14	8.01±0.07	0.1709	17.32±0.34	9.94±0.15
REBEL (random)	0.1496	16.84±0.14	8.14±0.08	0.1718	15.08±0.19	9.27±0.06
QRPO	0.1521	17.20±0.29	8.22±0.08	0.1748	16.32±0.29	9.07±0.18
QRPO (random)	**0.1522**	16.92±0.08	8.05±0.07	0.1695	15.31±0.30	8.54±0.24
BoNBoN	0.1409	12.74±0.05	5.92±0.04	0.1641	13.76±0.20	7.60±0.07
TUP mild	0.1500	17.47±0.28	8.38±0.12	0.1798	20.68±0.46	12.27±0.25
TUP mid.	0.1499	17.45±0.13	8.39±0.07	**0.1806**	**21.24±0.20**	**12.49±0.11**
TUP aggressive	0.1499	**17.54±0.16**	**8.43±0.05**	0.1794	19.64±0.58	11.38±0.24

3.3 Training objective: policy learning as a classification problem

Having established the theoretical properties of our proposed policy $\pi^\star_{\lambda,\beta}$, we now turn to formulate the training objective induced by this target. For completions in the support of $\pi^\star_{\lambda,\beta}$, rearranging (8) gives

$$w_{\lambda,r}(x,y) \;=\; \sigma\left(\beta \log \frac{\pi^\star_{\lambda,\beta}(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)} + \beta \log Z_{\lambda,\beta}\right), \tag{10}$$

where σ is the sigmoid function and logit is its inverse on $(0,1)$, so $\sigma(\mathrm{logit}(x)) = x$ for $x \in (0,1)$, with endpoint limits 0 and 1 as $x \to 0^+$ and $x \to 1^-$. The above relation suggests that we can fit a parameterized policy π_θ that estimates $\pi^\star_{\lambda,\beta}$ using logistic regression of the shifted-truncated win-rate. Indeed, we can define our classifier as

$$p_\theta(x,y) \;:=\; \sigma\left(\beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)} + \beta \log Z_{\lambda,\beta}\right), \tag{11}$$

where $\beta \log Z_{\lambda,\beta}$ acts as a constant and known intercept. For empirical finite-pool ranks, the normalizer has an exact finite-K analogue of the same form, with the integral replaced by a discrete average over rank locations. As in QRPO, our reported experiments use the population intercept, while Appendix A.1 gives the finite-pool alternative. Empirically, the population and finite-pool normalizers performs similarly to the analytical normalizer, except at extremely small β, where the finite-pool form is needed to keep the normalizer finite.

The construction above naturally results in a fully offline binary cross-entropy (BCE) objective with the shifted-truncated win-rate as a soft label:

The **BCE TUP distillation** objective is:

$$\mathcal{L}(\theta) \;=\; \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_{\mathrm{ref}}(\cdot|x)}\Big[-w_{\lambda,r}(x,y)\log\big(p_\theta(x,y)\big) - (1-w_{\lambda,r}(x,y))\log\big(1-p_\theta(x,y)\big)\Big]. \tag{12}$$

This loss requires only precomputed scalar rank labels $w_{\lambda,r}(x,y)$ and the global intercept $\beta \log Z_{\lambda,\beta}$. It requires no pairwise preferences, no online sampling, and no runtime estimation of a prompt-dependent partition function.

Algorithm 1 summarizes the resulting fully offline training procedure. In practice, for each prompt we first sample a reference pool, compute each completion's empirical in-pool win-rate $\hat{w}_r(x,y)$ against the other $K-1$ completions in that same pool, apply the global truncation λ, and then fit π_θ by binary cross-entropy on the distilled-to-reference log-likelihood ratio.

Table 2: AlpacaEval performance for Llama 8B Tülu 3 SFT trained on UltraFeedback (top) and Magpie Air (bottom). We report the LC reward metric for each reward model, and win-rate using `gpt-4o` judge results; length is measured in characters. Methods marked "random" are trained on a random pair of completions from each pool, whereas unmarked pairwise methods use the best and worst completions. TUP is trained on a single random completion per pool.

		Reward model			GPT Judge		
Train set	Method	ArmoRM	Skywork-Llama	Skywork-Qwen	LC Win	Win	Len
UltraFeedback	Initial	0.1448	13.44±1.60	6.65±0.89	15.18±0.36	8.57±0.98	1011
	DPO	0.1619	21.59±0.16	10.91±0.08	**42.30±0.16**	37.08±1.70	1807
	REBEL	0.1620	22.15±0.12	11.19±0.08	41.81±0.12	39.81±1.73	1938
	REBEL (random)	0.1625	22.50±0.18	11.31±0.05	**42.44±0.15**	40.93±1.73	1959
	QRPO	**0.1668**	20.69±0.19	10.23±0.13	39.64±0.16	**53.17±1.76**	3100
	QRPO (random)	0.1663	22.26±0.02	11.11±0.03	**42.35±0.13**	48.51±1.76	2436
	BoNBoN	0.1503	16.75±0.30	8.44±0.19	23.28±0.43	15.40±1.27	1325
	TUP mild	0.1638	22.85±0.37	11.41±0.15	39.41±0.14	49.50±1.76	2730
	TUP mid.	0.1633	**23.05±0.23**	**11.54±0.05**	40.27±0.06	49.13±1.76	2670
	TUP aggressive	0.1638	22.90±0.31	11.46±0.19	**42.36±0.14**	51.24±1.76	2709
Magpie Air	Initial	0.1448	13.44±1.60	6.65±0.89	15.18±0.36	8.57±0.98	1011
	DPO	0.1581	18.94±0.16	9.73±0.06	32.11±0.25	26.15±1.55	1644
	REBEL	0.1591	20.65±0.19	10.37±0.12	39.66±0.21	36.58±1.70	1872
	REBEL (random)	0.1565	17.81±0.14	9.15±0.07	31.37±0.26	26.21±1.55	1675
	QRPO	0.1578	19.00±0.08	9.30±0.00	32.39±0.13	32.48±1.65	1992
	QRPO (random)	0.1553	18.64±0.31	9.24±0.15	27.97±0.27	23.54±1.49	1628
	BoNBoN	0.1446	14.92±0.22	7.32±0.17	19.82±0.42	13.73±1.21	1346
	TUP mild	0.1585	20.58±0.36	10.21±0.15	36.82±0.02	**43.29±1.75**	2712
	TUP mid.	0.1585	**21.08±0.44**	**10.44±0.08**	35.81±0.04	42.55±1.74	2645
	TUP aggressive	0.1574	19.76±0.39	9.77±0.13	32.31±0.03	38.94±1.72	2626

4 Experiments

Experimental setup. We build on the QRPO experimental framework of Matrenok et al. [26], which evaluates offline alignment methods across multiple models and datasets. We compare TUP with six baselines across two model families, two datasets, three reward-model evaluators, and a GPT judge. We use UltraFeedback [7] and Magpie Air [40] as training datasets. In our experiments, we apply each alignment method to Llama 8B Tülu 3 SFT [22] separately on both datasets and to Mistral-7B [18] on Magpie Air; for Mistral, we first perform dedicated supervised fine-tuning (SFT) to obtain the initial reference policy.

Following QRPO, we use their published datasets, which use ArmoRM [39] as the offline reward model for scoring the reference-model training completions. To test generalization beyond the training reward model, we evaluate using two Skywork-v2 reward models, Skywork-Reward-V2-Llama-3.1-8B and Skywork-Reward-V2-Qwen3-8B [25], referred to as Skywork-Llama and Skywork-Qwen respectively. These models are ranked #1 and #2, respectively, among available classifier reward models on the RewardBench leaderboard at the time of writing [23].[1] We additionally evaluate on AlpacaEval using `gpt-4o` as a judge [16, 45]. Table 8 of the Appendix reports the RewardBench standings of all reward and judge models in our experiments. We report length-controlled (LC) rewards in the main tables; raw reward results are given in Appendix B.1.

The most natural baselines for our setting are rank-based distillation methods. We include BoNBoN [14] and the QRPO variant that computes the loss with random completions [26], denoted QRPO (random). While less related, we also compare to strong offline-alignment baselines using pairwise or relative-reward signals, namely DPO [31] and REBEL [12]. Unless marked random, DPO and REBEL use the best–worst pair from each pool; REBEL (random) uses random pairs. We also include QRPO trained on best–worst pairs. For all DPO, REBEL, and QRPO variants, we adopt the best-performing configurations from the *off-policy* setting reported in Matrenok et al. [26]. To avoid clutter, tables omit the best–worst label when it is the default. We do not include SimPO [27], whose length-normalized objective makes it structurally different from baselines that do not explicitly

[1] `https://huggingface.co/spaces/allenai/reward-bench`

Table 3: LC rewards for Mistral 7B SFT, trained on Magpie Air. Tested on in-dataset test split (MA) and AlpacaEval (AE) evaluated by ArmoRM, Skywork-Llama and Skywork-Qwen reward models, with AlpacaEval also judged by `gpt-4o`.

	ArmoRM		Skywork-v2-Llama		Skywork-v2-Qwen		GPT Judge		
Method	MA	AE	MA	AE	MA	AE	LC Win	Win	Len
Initial	0.1646	0.1477	17.36±0.11	16.53±0.19	11.57±0.07	8.76±0.09	21.94±0.27	18.76±1.38	1736
DPO	0.1792	0.1583	22.62±0.22	21.50±0.11	16.94±0.19	12.43±0.05	31.80±0.05	37.14±1.70	2860
REBEL	0.1732	0.1462	20.83±0.16	13.58±0.19	14.08±0.15	7.51±0.15	22.11±0.02	19.25±1.39	8470
REBEL (random)	0.1822	0.1588	**24.64±0.23**	**21.91±0.46**	**18.32±0.26**	**12.67±0.18**	34.45±0.12	40.12±1.73	3013
QRPO	0.1720	0.1565	21.09±0.12	20.29±0.41	14.23±0.03	11.02±0.14	31.97±0.19	34.78±1.68	2261
QRPO (random)	0.1858	0.1425	21.96±0.22	17.23±1.53	15.86±0.20	9.54±1.04	27.53±0.12	29.07±1.60	4740
BoNBoN	0.1667	0.1494	18.40±0.32	17.46±0.13	12.35±0.35	9.34±0.09	24.85±0.21	22.11±1.46	1828
TUP mild	**0.1864**	0.1612	23.47±0.15	21.04±0.56	17.01±0.10	11.75±0.35	**36.84±0.04**	**40.62±1.73**	3382
TUP mid.	0.1832	0.1620	22.81±0.18	21.34±0.34	16.96±0.08	12.25±0.11	35.69±0.05	39.25±1.72	2765
TUP aggressive	0.1826	**0.1625**	21.32±0.12	20.50±0.76	16.00±0.02	11.54±0.34	33.22±0.14	39.63±1.73	2688

account for length. Because BoNBoN is not covered by the QRPO benchmark, we run a dedicated hyperparameter search and select the best checkpoint by validation reward. Full implementation details are provided in Appendix C.

In all tables, unless stated otherwise, we show TUP with 3 different global truncation values; $\lambda = 0.2$ (mild) which truncates only the worst rollout out of the 6 completions, $\lambda = 0.5$ (mid.) and $\lambda = 0.8$ (aggressive) that keeps only the top two completions. All results report standard errors, except for ArmoRM, whose standard errors are negligible (only affecting the fourth decimal place) and are omitted for legibility.

Results. Table 1 reports two in-dataset experiments on Llama 8B, where each method is evaluated on the held-out split of its respective training dataset, UltraFeedback [7] or Magpie Air [40]. As shown, TUP is competitive with strong offline alignment baselines on both datasets. Under ArmoRM, all methods achieve broadly similar results. Larger differences appear under the two Skywork reward models. On both UltraFeedback and Magpie Air, TUP obtains the best in-dataset results under both Skywork-Llama and Skywork-Qwen. If these reward models are viewed as different proxies for the oracle reward, the results suggest that TUP generalizes well across proxy rewards compared to the baselines considered.

Table 2 reports AlpacaEval results for the above mentioned trained Llama models. Under both datasets, TUP obtains the best results under both Skywork-Llama and Skywork-Qwen, including on AlpacaEval LC reward, and remains highly competitive under the GPT judge. While the Magpie-Air-trained models on AlpacaEval results are more mixed, TUP remains competitive. The supplementary raw reward tables in Appendix B.1 provide the corresponding non-LC scores and in-dataset response lengths, showing that the main trends are not limited to the LC metric. Table 3, which reports both in-dataset and AlpacaEval LC rewards, shows that TUP remains competitive when applied to the Mistral model family.

In addition to the LC-reward results reported above, we also conduct an experiment to further isolate TUP's reward gains from the effect of its longer average responses. This experiment compares rewards for length-matched response pairs between TUP and each baseline. As shown in Table 7 in the Appendix, TUP responses are largely preferred over length-matched baseline responses to the same prompt. This suggests that TUP's advantage does not stem from response length alone. Full details are in Section B.2 of the Appendix.

Figure 4 shows the effects of TUP's truncation threshold λ and preference sharpness β, using the same validation-based checkpoint-selection protocol as the main experiments (Tables 1–5). To approximate the no-truncation setting, we use $\lambda = 0.2$, which removes only the lowest-ranked of the six responses, a response that would otherwise receive an extremely low weight. To approximate "pure" truncation, we use $\beta = 100$, which produces an approximately uniform weighting over the retained tail while still remaining finite and allowing for some deviation from π_{ref}. Generally, intermediate values of both λ and β yield the highest LC rewards. These results suggest that the best performance comes from combining moderate truncation with non-uniform within-tail reweighting, supporting our choice to treat λ and β as separate parameters.

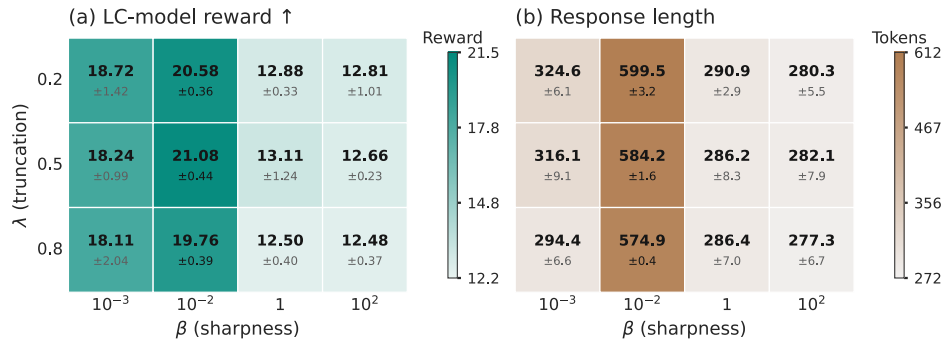

Figure 4: Ablation study of TUP's truncation threshold λ and sharpness β for Llama 8B Tülu 3 SFT trained on Magpie Air and evaluated on AlpacaEval with Skywork-Llama reward model. TUP uses $K = 6$ reference completions per prompt. Cells report the mean and sample standard error across three generation seeds. **Left:** LC rewards; higher is better. **Right:** Mean response length in tokens.

5 Discussion

One limitation of our method is that both λ and β must be tuned based on validation performance, whereas other baselines tune only β. This may increase the computational cost when a suitable initial range for λ is unknown. A practical way to narrow this range is to construct a plot such as the right panel of Figure 2. Given a dataset, a proxy reward model, and an auxiliary reward model treated as an "oracle," this plot provides an efficient way to identify a promising range of truncation levels. Once a truncation level that retains responses ranked highly by the "oracle" while removing poorly ranked ones is identified, the remaining hyperparameter search is once again only on β values.

TUP also leaves several directions for future work. First, although TUP improves rank-based policy distillation, it is still exposed to reward hacking; incorporating reward-hacking mitigation methods could further improve its performance. Second, future work could replace the fixed global λ and β with potentially learned prompt-specific parameters.

More broadly, TUP may support safety alignment when unsafe or otherwise undesirable completions are consistently assigned low proxy ranks, allowing truncation to remove them from the target support rather than merely down-weight them.

Acknowledgments and Disclosure of Funding

This research was supported by the European Union (ERC, SafetyBounds, 101163414). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan Daniel Chang, and Prithviraj Ammanabrolu. Critique-out-Loud Reward Models. In *Pluralistic Alignment Workshop at NeurIPS*, 2024.

[2] 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.

[3] Ananth Balashankar, Ziteng Sun, Jonathan Berant, Jacob Eisenstein, Michael Collins, Adrian Hutter, Jong Lee, Chirag Nagpal, Flavien Prost, and Aradhana Sinha. InfAlign: Inference-aware language model alignment. In *Forty-second International Conference on Machine Learning*, 2025.

[4] Ahmad Beirami, Alekh Agarwal, Jonathan Berant, Alexander D'Amour, Jacob Eisenstein, Chirag Nagpal, and Ananda Theertha Suresh. Theoretical guarantees on the best-of-N alignment policy. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

[5] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[6] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, 2017.

[7] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback. 2023.

[8] Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, KaShun Shum, and Tong Zhang. RAFT: Reward rAnked FineTuning for Generative Foundation Model Alignment. *Transactions on Machine Learning Research*, 2023.

[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] Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios Nikolas Angelopoulos, Jiantao Jiao, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. How to Evaluate Reward Models for RLHF. In *The Thirteenth International Conference on Learning Representations*, 2025.

[11] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[12] Zhaolin Gao, Jonathan D. Chang, Wenhao Zhan, Owen Oertell, Gokul Swamy, Kianté Brantley, Thorsten Joachims, J. Andrew Bagnell, Jason Lee, and Wen Sun. REBEL: Reinforcement Learning via Regressing Relative Rewards. In *Advances in Neural Information Processing Systems*, 2024.

[13] Shivank Garg, Ayush Singh, Shweta Singh, and Paras Chopra. IPO: Your Language Model is Secretly a Preference Classifier. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

[14] Lin Gui, Cristina Gârbacea, and Victor Veitch. BoNBoN alignment for large language models and the sweetness of best-of-n sampling. In *Advances in Neural Information Processing Systems*, 2024.

[15] Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan J Foster. Is Best-of-N the Best of Them? Coverage, Scaling, and Optimality in Inference-Time Alignment. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

[16] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[17] Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A. Smith, Yejin Choi, and Hannaneh Hajishirzi. Unpacking DPO and PPO: Disentangling Best Practices for Learning from Preference Feedback. In *Advances in Neural Information Processing Systems*, 2024.

[18] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[19] Yuu Jinnai, Tetsuro Morimura, Kaito Ariu, and Kenshi Abe. Regularized best-of-n sampling to mitigate reward hacking for language model alignment. In *ICML 2024 Workshop on Models of Human Feedback for AI Alignment*, 2024.

[20] Hadi Khalaf, Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio du Pin Calmon. Inference-Time Reward Hacking in Large Language Models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[21] Sunghwan Kim, Dongjin Kang, Taeyoon Kwon, Hyungjoo Chae, Dongha Lee, and Jinyoung Yeo. Rethinking reward model evaluation through the lens of reward overoptimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

[22] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025.

[23] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, Lester James Validad Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, and Yejin Choi. RewardBench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, 2025.

[24] Eddie Landesberg. When LLM judge scores look good but best-of-n decisions fail. *arXiv preprint arXiv:2603.12520*, 2026.

[25] Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, and Jiacheng Xu. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. In *The Fourteenth International Conference on Learning Representations*, 2026.

[26] Simon Matrenok, Skander Moalla, and Caglar Gulcehre. Quantile Reward Policy Optimization: Alignment with Pointwise Regression and Exact Partition Functions. In *Advances in Neural Information Processing Systems*, 2025.

[27] Yu Meng, Mengzhou Xia, and Danqi Chen. SimPO: Simple preference optimization with a reference-free reward. In *Advances in Neural Information Processing Systems*, 2024.

[28] Eric J Michaud, Adam Gleave, and Stuart Russell. Understanding learned reward functions. *arXiv preprint arXiv:2012.05862*, 2020.

[29] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[30] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, and Alex Ray. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 2022.

[31] 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. In *Advances in Neural Information Processing Systems*, 2023.

[32] Rafael Rafailov, Yaswanth Chittepu, Ryan Park, Harshit Sikchi, Joey Hejna, W. Bradley Knox, Chelsea Finn, and Scott Niekum. Scaling laws for reward model overoptimization in direct alignment algorithms. In *Advances in Neural Information Processing Systems*, 2024.

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

[34] Pier Giuseppe Sessa, Robert Dadashi-Tazehozi, Leonard Hussenot, Johan Ferret, Nino Vieillard, Alexandre Rame, Bobak Shahriari, Sarah Perrin, Abram L Friesen, and Geoffrey Cideron. BOND: Aligning LLMs with Best-of-N Distillation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[35] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[36] Yifan Song, Guoyin Wang, Sujian Li, and Bill Yuchen Lin. The Good, The Bad, and The Greedy: Evaluation of LLMs Should Not Ignore Non-Determinism. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2025.

[37] 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. In *Advances in Neural Information Processing Systems*, 2020.

[38] Jeremy Tien, Jerry Zhi-Yang He, Zackory Erickson, Anca Dragan, and Daniel S. Brown. Causal confusion and reward misidentification in preference-based reward learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[39] Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.

[40] Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned LLMs with nothing. In *The Thirteenth International Conference on Learning Representations*, 2025.

[41] Tong Yang, Jincheng Mei, Hanjun Dai, Zixin Wen, Shicong Cen, Dale Schuurmans, Yuejie Chi, and Bo Dai. Faster wind: Accelerating iterative best-of-n distillation for LLM alignment. In *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, 2025.

[42] 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, 2023.

[43] Junkai Zhang, Zihao Wang, Lin Gui, Swarnashree Mysore Sathyendra, Jaehwan Jeong, Victor Veitch, Wei Wang, Yunzhong He, Bing Liu, and Lifeng Jin. Chasing the Tail: Effective Rubric-based Reward Modeling for Large Language Model Post-Training. In *The Fifteenth International Conference on Learning Representations*, 2026.

[44] Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. SLIC-HF: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

[45] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

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

A Theory and proofs

A.1 Normalization constant for Proposition 3.1

Proof. By the Gibbs solution in (2), substituting $r = R_\lambda$ yields

$$\pi^\star(y \mid x) \;=\; \frac{1}{Z(x)}\,\pi_{\mathrm{ref}}(y \mid x)\exp\!\left(\frac{R_\lambda(x,y)}{\beta}\right) \;=\; \frac{1}{Z(x)}\,\pi_{\mathrm{ref}}(y \mid x)\left(\frac{w_{\lambda,r}(x,y)}{1-w_{\lambda,r}(x,y)}\right)^{1/\beta}.$$

It therefore remains to identify the normalizer $Z(x)$ and show that it is independent of x. Under the induced uniform win-rate variable $w \sim \mathrm{Uniform}[0,1]$, the normalization term becomes

$$Z(x) \;=\; \int_0^1 \left(\frac{\max(w-\lambda,\,0)}{1-\max(w-\lambda,\,0)}\right)^{1/\beta} dw. \tag{13}$$

For $w \le \lambda$, the shifted win-rate is zero, so the integrand vanishes. Therefore,

$$Z(x) \;=\; \int_\lambda^1 \left(\frac{w-\lambda}{1-(w-\lambda)}\right)^{1/\beta} dw. \tag{14}$$

Applying the change of variables $u := w - \lambda$ gives

$$Z(x) \;=\; \int_0^{1-\lambda} \left(\frac{u}{1-u}\right)^{1/\beta} du \;=\; \int_0^{1-\lambda} u^{\,1/\beta}(1-u)^{-1/\beta}\,du. \tag{15}$$

Recalling that the incomplete Beta function is defined in Proposition 3.1 as $B_z(a,b) = \int_0^z t^{a-1}(1-t)^{b-1}\,dt$, we identify

$$a = 1 + \tfrac{1}{\beta}, \qquad b = 1 - \tfrac{1}{\beta}.$$

Hence

$$Z(x) \;=\; \int_0^{1-\lambda} u^{\,(1+1/\beta)-1}(1-u)^{\,(1-1/\beta)-1}\,du \;=\; \mathrm{Beta}_{1-\lambda}\!\Big(1+\tfrac{1}{\beta},\,1-\tfrac{1}{\beta}\Big) \;=\; Z_{\lambda,\beta}, \tag{16}$$

which is independent of x. □

Finite-pool normalizer. The closed-form normalizer in Proposition 3.1 is the population normalizer induced by the continuous win-rate $w_r(x,Y)$. In the experiments, labels are computed from finite prompt-specific pools. Under the empirical-rank convention in Algorithm 1, the corresponding discrete prompt-independent normalizer is

$$Z_{K,\lambda,\beta} = \frac{1}{K}\sum_{j=1}^{K}\left(\frac{(\frac{j}{K}-\lambda)_+}{1-(\frac{j}{K}-\lambda)_+}\right)^{1/\beta}.$$

This finite-pool form can replace $Z_{\lambda,\beta}$ in the BCE intercept without changing the labels, the truncation support, or the loss. In our experiments, we use the population normalizer $Z_{\lambda,\beta}$, following the continuous win-rate normalization used in QRPO; in our experiments, replacing it with the finite-pool expression led to negligible performance differences. In extremely small-β regimes, where the continuous calculation can become numerically unstable, the discrete expression above provides the exact finite-K alternative for this empirical-rank convention.

A.2 Formal oracle rank-space results for Section 3.2

This appendix formalizes the oracle rank-space discussion from Section 3.2. The aim is to separate two decisions that smooth full-support reweighting conflates: which proxy ranks should remain in the support at all, and how the retained ranks should be reweighted. The first result shows that, among monotone proxy-rank reweightings, the oracle-best rule is always a hard lower-tail truncation. The second result is a fixed-threshold refinement statement: once a truncation level has been chosen, finite within-tail reweighting can improve over pure truncation when oracle value remains positively aligned with proxy rank inside the retained tail. We close with the quadratic worked example behind Figure 3.

Fix a prompt x, and let $Y \sim \pi_{\mathrm{ref}}(\cdot \mid x)$. Define the proxy and oracle population win-rates by

$$w_r(x,y) := \mathbb{P}_{Y'\sim\pi_{\mathrm{ref}}(\cdot|x)}\big(r(x,Y') \le r(x,y)\big), \qquad w_u(x,y) := \mathbb{P}_{Y'\sim\pi_{\mathrm{ref}}(\cdot|x)}\big(u(x,Y') \le u(x,y)\big).$$

and write

$$W_r := w_r(x, Y), \qquad W_u := w_u(x, Y).$$

Since BoN-style methods act on relative rank rather than raw reward, the natural policy class in this analysis reweights a completion according to its proxy win-rate. For any measurable $g : [0, 1] \to [0, \infty)$ satisfying

$$0 < \mathbb{E}_{\pi_{\mathrm{ref}}}[g(W_r)] < \infty,$$

define

$$\pi_g(y \mid x) := \frac{g(w_r(x, y))\, \pi_{\mathrm{ref}}(y \mid x)}{\mathbb{E}_{Y' \sim \pi_{\mathrm{ref}}(\cdot|x)}[g(w_r(x, Y'))]}, \qquad \mathcal{U}_x(g) := \mathbb{E}_{Y \sim \pi_g(\cdot|x)}[W_u].$$

Finally, let

$$m_x(w) := \mathbb{E}[W_u \mid W_r = w]$$

denote a measurable version of the conditional oracle profile. Once the problem is reduced to rank space, m_x is the only prompt-specific object that matters.

A.2.1 Adaptive truncation among monotone reweightings

We first isolate the support-selection question of which proxy ranks should receive positive mass at all.

Lemma A.1 (Exact reduction to proxy-rank space)**.** *For every measurable g as above,*

$$\mathcal{U}_x(g) = \frac{\mathbb{E}_{\pi_{\mathrm{ref}}}[m_x(W_r)g(W_r)]}{\mathbb{E}_{\pi_{\mathrm{ref}}}[g(W_r)]}.$$

If moreover the law of $r(x, Y)$ under $Y \sim \pi_{\mathrm{ref}}(\cdot \mid x)$ is continuous, then $W_r \sim \mathrm{Unif}[0, 1]$, and therefore

$$\mathcal{U}_x(g) = \frac{\int_0^1 m_x(w)g(w)\, dw}{\int_0^1 g(w)\, dw}.$$

Proof. By definition of π_g,

$$\mathcal{U}_x(g) = \frac{\mathbb{E}_{\pi_{\mathrm{ref}}}[W_u g(W_r)]}{\mathbb{E}_{\pi_{\mathrm{ref}}}[g(W_r)]}.$$

Applying conditional expectation with respect to W_r gives

$$\mathbb{E}_{\pi_{\mathrm{ref}}}[W_u g(W_r)] = \mathbb{E}_{\pi_{\mathrm{ref}}}[\mathbb{E}[W_u g(W_r) \mid W_r]] = \mathbb{E}_{\pi_{\mathrm{ref}}}[g(W_r)\mathbb{E}[W_u \mid W_r]] = \mathbb{E}_{\pi_{\mathrm{ref}}}[m_x(W_r)g(W_r)].$$

This proves the first identity. If the law of $r(x, Y)$ is continuous, the probability integral transform yields $W_r \sim \mathrm{Unif}[0, 1]$, and the second identity follows by writing expectations as Lebesgue integrals on $[0, 1]$. □

Under the continuity assumption, the problem becomes a pure rank-space optimization over densities on $[0, 1]$. Let

$$\mathcal{F}_\uparrow := \left\{ f : [0, 1] \to [0, \infty) : \int_0^1 f(w)\, dw = 1, \;\; f \text{ is nondecreasing} \right\},$$

and for each threshold $\lambda \in [0, 1)$, define the lower-tail truncation density

$$\nu_\lambda(w) := \frac{1}{1 - \lambda} \mathbb{1}\{w \in [\lambda, 1]\}.$$

The next lemma shows that every monotone density is a mixture of these top-interval uniforms. This is the structural reason hard truncation is enough.

Lemma A.2 (Mixture representation of nondecreasing densities)**.** *Let $f \in \mathcal{F}_\uparrow$. Then there exists a probability measure ξ on $[0, 1)$ such that*

$$f(w) = \int_{[0,1)} \nu_\lambda(w)\, \xi(d\lambda) \qquad \text{for a.e. } w \in [0, 1].$$

Proof. Define a finite Stieltjes measure on $[0,1)$ by

$$\xi := f(0)\delta_0 + (1-t)\,df(t).$$

Since f is nondecreasing and $\int_0^1 f(w)\,dw = 1$, integration by parts gives

$$\xi([0,1)) = f(0) + \int_{(0,1)} (1-t)\,df(t) = 1.$$

Thus ξ is a probability measure. For a.e. $w \in [0,1]$,

$$\int_{[0,1)} \nu_\lambda(w)\,\xi(d\lambda) = \int_{[0,w]} \frac{1}{1-\lambda}\,\xi(d\lambda) = f(0) + \int_{(0,w]} df(\lambda) = f(w).$$

□

Theorem A.3 (Hard truncation is optimal among monotone reweightings)**.** *Assume the law of $r(x,Y)$ under $Y \sim \pi_{\mathrm{ref}}(\cdot \mid x)$ is continuous. Then*

$$\sup_{f \in \mathcal{F}_\uparrow} \mathcal{U}_x(f) = \sup_{\lambda \in [0,1)} \mathcal{U}_x(\nu_\lambda).$$

Equivalently, in the oracle rank-space view, optimizing over monotone proxy-rank reweightings reduces to optimizing over hard lower-tail truncation rules.

Proof. By Lemma A.1, the continuity assumption implies

$$\mathcal{U}_x(f) = \int_0^1 m_x(w) f(w)\,dw \qquad \text{for every } f \in \mathcal{F}_\uparrow.$$

Fix $f \in \mathcal{F}_\uparrow$. By Lemma A.2,

$$f(w) = \int_{[0,1)} \nu_\lambda(w)\,\xi(d\lambda) \qquad \text{for a.e. } w,$$

for some probability measure ξ on $[0,1)$. Therefore,

$$\mathcal{U}_x(f) = \int_0^1 m_x(w) \left(\int_{[0,1)} \nu_\lambda(w)\,\xi(d\lambda) \right) dw.$$

By Fubini's theorem,

$$\mathcal{U}_x(f) = \int_{[0,1)} \left(\int_0^1 m_x(w)\nu_\lambda(w)\,dw \right) \xi(d\lambda) \le \sup_{\lambda \in [0,1)} \mathcal{U}_x(\nu_\lambda).$$

Taking the supremum over $f \in \mathcal{F}_\uparrow$ gives

$$\sup_{f \in \mathcal{F}_\uparrow} \mathcal{U}_x(f) \le \sup_{\lambda \in [0,1)} \mathcal{U}_x(\nu_\lambda).$$

The reverse inequality is immediate because each ν_λ belongs to $\mathcal{F}_\uparrow$. □

Theorem A.3 is an oracle statement, so it does not identify a single global threshold for practice. Its role is structural: in rank space, support restriction is the right family for the first decision, namely which proxy ranks to keep.

Corollary A.4 (QRPO comparison in the same rank-space model)**.** *Under the assumptions of Theorem A.3, let*

$$q_\tau(w) := \frac{\tau e^{\tau w}}{e^\tau - 1}, \qquad \tau > 0,$$

with $q_0(w) \equiv 1$. Then

$$\sup_{\tau \ge 0} \mathcal{U}_x(q_\tau) \le \sup_{\lambda \in [0,1)} \mathcal{U}_x(\nu_\lambda).$$

Proof. Each q_τ is nondecreasing on $[0,1]$, so $\{q_\tau : \tau \ge 0\} \subseteq \mathcal{F}_\uparrow$. The claim follows immediately from Theorem A.3. □

Thus, within the same oracle rank-space model, the best hard truncation rule weakly dominates the best QRPO-style full-support monotone tilt.

A.2.2 Within-tail reweighting after truncation

We now hold the truncation level fixed and show that, when oracle value remains positively aligned with proxy rank inside the retained tail, uniform weighting over the retained interval is locally suboptimal, so a finite within-tail tilt can improve over pure truncation at the same threshold.

For a fixed threshold $\lambda \in (0,1)$ and parameter $\beta \in (0,\infty)$, define the truncated-soft family

$$f_{\lambda,\beta}(w) := \frac{1}{Z_\lambda(\beta)} \left(\frac{w-\lambda}{1-w+\lambda} \right)^{1/\beta} \mathbb{1}\{w \in (\lambda,1)\},$$

where

$$Z_\lambda(\beta) := \int_\lambda^1 \left(\frac{w-\lambda}{1-w+\lambda} \right)^{1/\beta} dw.$$

This is the rank-space form induced by the shifted-truncated odds-ratio tilt at threshold λ. We use the convention $f_{\lambda,\infty} = \nu_\lambda$, so $\beta = \infty$ recovers hard truncation, while finite β tilts mass toward larger retained proxy ranks without changing the support. Define

$$\mathcal{U}_x(\lambda,\beta) := \int_\lambda^1 m_x(w) f_{\lambda,\beta}(w)\, dw, \qquad \mathcal{U}_x(\lambda,\infty) := \mathcal{U}_x(\nu_\lambda).$$

Proposition A.5 (Local criterion for improving over pure truncation)**.** *Assume the law of $r(x,Y)$ under $Y \sim \pi_{\mathrm{ref}}(\cdot \mid x)$ is continuous, and fix $\lambda \in (0,1)$. Let*

$$h_\lambda(w) := \log\!\left(\frac{w-\lambda}{1-w+\lambda} \right), \qquad w \in (\lambda,1).$$

Then

$$\frac{\partial}{\partial(\beta^{-1})} \mathcal{U}_x(\lambda,\beta)\Big|_{\beta=\infty} = \mathrm{Cov}_{W_r \sim \mathrm{Unif}[\lambda,1]}\big(m_x(W_r),\, h_\lambda(W_r)\big).$$

In particular, if this covariance is positive, then there exists a finite β such that

$$\mathcal{U}_x(\lambda,\beta) > \mathcal{U}_x(\lambda,\infty) = \mathcal{U}_x(\nu_\lambda).$$

Proof. Let $f_{\lambda,\infty} = \nu_\lambda$ denote the uniform density on $[\lambda,1]$. Then

$$f_{\lambda,\beta}(w) = \frac{e^{h_\lambda(w)/\beta} f_{\lambda,\infty}(w)}{\mathbb{E}_{W_r \sim \mathrm{Unif}[\lambda,1]}\big[e^{h_\lambda(W_r)/\beta}\big]}.$$

Therefore,

$$\mathcal{U}_x(\lambda,\beta) = \frac{\mathbb{E}_{W_r \sim \mathrm{Unif}[\lambda,1]}\big[m_x(W_r) e^{h_\lambda(W_r)/\beta}\big]}{\mathbb{E}_{W_r \sim \mathrm{Unif}[\lambda,1]}\big[e^{h_\lambda(W_r)/\beta}\big]}.$$

Differentiating with respect to β^{-1} at $\beta = \infty$ gives

$$\frac{\partial}{\partial(\beta^{-1})} \mathcal{U}_x(\lambda,\beta)\Big|_{\beta=\infty} = \mathbb{E}[m_x(W_r) h_\lambda(W_r)] - \mathbb{E}[m_x(W_r)]\,\mathbb{E}[h_\lambda(W_r)],$$

where all expectations are under $W_r \sim \mathrm{Unif}[\lambda,1]$. The differentiation may be passed under the expectation because h_λ is integrable on $(\lambda,1)$ under the uniform law. This is exactly

$$\mathrm{Cov}_{W_r \sim \mathrm{Unif}[\lambda,1]}\big(m_x(W_r),\, h_\lambda(W_r)\big).$$

If this covariance is positive, then the derivative at $\beta = \infty$ in the β^{-1} direction is positive, so by continuity there exists a finite β such that

$$\mathcal{U}_x(\lambda,\beta) > \mathcal{U}_x(\lambda,\infty).$$

□

This is a fixed-λ refinement statement, not a global dominance claim. It does not say that finite-β reweighting beats the best threshold when λ is optimized freely. Instead, once a global quality floor has been chosen, the covariance criterion identifies when a softer within-tail tilt extracts additional oracle value from the retained tail.

A.2.3 Quadratic illustration underlying Figure 3

This subsection gives a minimal analytic example that separates the two effects shown in Figure 3: changing the retained support and reweighting points within a fixed retained tail. Consider the stylized oracle-utility shape

$$m_s(w) := 1 - (w - s)^2, \qquad s \in (0, 1),$$

where s is the proxy-rank location at which oracle value is maximized. We use m_s only as an unnormalized analytic shape. It is not itself a realizable conditional oracle win-rate profile, since

$$\int_0^1 m_s(w)\, dw = \frac{2}{3} + s - s^2,$$

which is generally not $1/2$. A normalized profile with the same shape is

$$\bar{m}_s(w) := \frac{1}{2} + \frac{1}{2}\left(m_s(w) - \left(\frac{2}{3} + s - s^2\right)\right).$$

Then $\int_0^1 \bar{m}_s(w)\, dw = 1/2$, and $\bar{m}_s(w) \in [1/6, 2/3]$ for all $s, w \in [0, 1]$. For any density f on $[0, 1]$,

$$\bar{\mathcal{U}}_s(f) := \int_0^1 \bar{m}_s(w) f(w)\, dw = \frac{1}{2} + \frac{1}{2}\left(\mathcal{U}_s(f) - \left(\frac{2}{3} + s - s^2\right)\right),$$

where

$$\mathcal{U}_s(f) := \int_0^1 m_s(w) f(w)\, dw.$$

Thus, for each fixed s, $\bar{\mathcal{U}}_s$ is a positive affine transformation of $\mathcal{U}_s$. All maximizers, strict comparisons, crossing locations, and the threshold s_β below are unchanged by this normalization. We therefore use the shorter unnormalized form m_s in the algebra.

For the QRPO-style retained-tail family, write

$$\mathcal{U}_s(\lambda, \beta) := \int_\lambda^1 m_s(w) f_{\lambda,\beta}(w)\, dw, \qquad \mathcal{U}_s(\lambda, \infty) := \mathcal{U}_s(\nu_\lambda).$$

The first proposition isolates the value of adaptive support restriction by comparing the best hard truncation rule with the best full-support QRPO tilt. The second proposition fixes $\lambda = 1/2$, matching Figure 3, and asks when finite within-tail reweighting improves over pure truncation at that same threshold.

Proposition A.6 (Adaptive truncation beats best QRPO under the quadratic profile)**.** *For every $s > 1/2$, the hard truncation threshold*

$$\lambda^\star(s) := \frac{3s - 1}{2}$$

satisfies

$$\mathcal{U}_s(\nu_{\lambda^\star(s)}) = 1 - \frac{(1 - s)^2}{4} > \sup_{\tau \geq 0} \mathcal{U}_s(q_\tau).$$

Proof. Since $m_s(w) = 1 - (w - s)^2$, maximizing $\mathcal{U}_s(f)$ is equivalent to minimizing

$$R(f; s) := \mathbb{E}_{W_r \sim f}\left[(W_r - s)^2\right].$$

For the top-interval uniform law ν_λ, we have

$$\mathbb{E}_{\nu_\lambda}[W_r] = \frac{1 + \lambda}{2}, \qquad \mathrm{Var}_{\nu_\lambda}(W_r) = \frac{(1 - \lambda)^2}{12}.$$

Therefore

$$R(\nu_\lambda; s) = \left(\frac{1 + \lambda}{2} - s\right)^2 + \frac{(1 - \lambda)^2}{12}.$$

Differentiating with respect to λ gives

$$\frac{d}{d\lambda} R(\nu_\lambda; s) = \frac{2\lambda}{3} + \frac{1}{3} - s.$$

Thus the unique minimizer over $\lambda \in [0,1]$ is

$$\lambda^\star(s) = \frac{3s-1}{2},$$

which lies in $(0,1)$ for $s \in (1/2,1)$. Substituting this value gives

$$R(\nu_{\lambda^\star(s)}; s) = \frac{(1-s)^2}{4}, \qquad \mathcal{U}_s(\nu_{\lambda^\star(s)}) = 1 - \frac{(1-s)^2}{4}.$$

By Lemma A.2, every $f \in \mathcal{F}_\uparrow$ is a mixture of the top-interval uniform laws ν_λ. Since $\lambda \mapsto \mathcal{U}_s(\nu_\lambda)$ has the unique maximizer $\lambda^\star(s)$, the unique maximizer over $\mathcal{F}_\uparrow$ is $\nu_{\lambda^\star(s)}$. Hence

$$\sup_{\tau \geq 0} \mathcal{U}_s(q_\tau) \leq \mathcal{U}_s(\nu_{\lambda^\star(s)}).$$

It remains only to rule out equality. No finite-τ QRPO density equals $\nu_{\lambda^\star(s)}$: q_0 is uniform on $[0,1]$, while q_τ is strictly positive on all of $[0,1]$ for every $\tau > 0$, whereas $\nu_{\lambda^\star(s)}$ has zero density below $\lambda^\star(s) > 0$. Moreover, since m_s is bounded and continuous and q_τ converges weakly to a point mass at $w = 1$ as $\tau \to \infty$,

$$\lim_{\tau \to \infty} \mathcal{U}_s(q_\tau) = m_s(1) = 1 - (1-s)^2.$$

This limiting value is strictly smaller than

$$1 - \frac{(1-s)^2}{4} = \mathcal{U}_s(\nu_{\lambda^\star(s)}).$$

Therefore neither a finite QRPO tilt nor its limiting point mass at $w = 1$ attains the hard-truncation optimum, and the desired strict inequality follows. □

This comparison isolates the gain from support restriction itself. Even in the quadratic example, a full-support QRPO tilt cannot recover the oracle value achieved by the best hard truncation rule.

Proposition A.7 (At $\lambda = 1/2$, finite β helps above an explicit threshold)**.** *Define*

$$s_\beta := \frac{\frac{11}{12} - \log 2}{1 - \log 2} \approx 0.7284.$$

Then

$$\frac{\partial}{\partial(\beta^{-1})} \mathcal{U}_s(1/2, \beta)\Big|_{\beta=\infty} = (1 - \log 2)s - \left(\frac{11}{12} - \log 2\right).$$

In particular, if $s > s_\beta$, then there exists a finite β such that

$$\mathcal{U}_s(1/2, \beta) > \mathcal{U}_s(1/2, \infty) = \mathcal{U}_s(\nu_{1/2}).$$

Proof. We instantiate Proposition A.5 at the fixed threshold $\lambda = 1/2$. On the retained interval $[1/2, 1]$, the local log-odds tilt is

$$h(w) := \log\left(\frac{w - \frac{1}{2}}{\frac{3}{2} - w}\right), \qquad w \in (1/2, 1).$$

Proposition A.5 gives

$$\frac{\partial}{\partial(\beta^{-1})} \mathcal{U}_s(1/2, \beta)\Big|_{\beta=\infty} = \mathrm{Cov}_{W_r \sim \mathrm{Unif}[1/2,1]}\big(m_s(W_r), h(W_r)\big).$$

Since $m_s(W_r) = 1 - (W_r - s)^2$, this is

$$\frac{\partial}{\partial(\beta^{-1})} \mathcal{U}_s(1/2, \beta)\Big|_{\beta=\infty} = -\,\mathrm{Cov}_{W_r \sim \mathrm{Unif}[1/2,1]}\big((W_r - s)^2, h(W_r)\big).$$

The required covariance is explicit. Under $W_r \sim \mathrm{Unif}[1/2, 1]$,

$$\mathbb{E}[h(W_r)] = 2\int_{1/2}^{1} \log\left(\frac{w - \frac{1}{2}}{\frac{3}{2} - w}\right) dw = -\log 4,$$

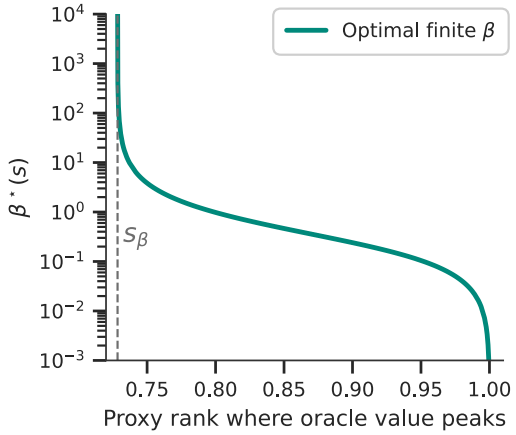

Figure 5: Optimal finite sharpness in the fixed-threshold quadratic illustration. For $\lambda = 1/2$ and $m_s(w) = 1 - (w - s)^2$, we numerically maximize $\mathcal{U}_s(1/2, \beta)$ over finite β for each oracle peak location $s > s_\beta$. The dashed vertical line marks $s_\beta \approx 0.7284$, below which pure truncation is locally optimal and the optimizer is $\beta = \infty$. Just above s_β, the optimal finite tilt is arbitrarily weak, so $\beta^\star(s)$ is large; as s approaches one, the optimal policy concentrates more strongly near the top proxy ranks, and $\beta^\star(s)$ decreases.

and

$$\mathbb{E}[(W_r - s)^2] = \left(s - \frac{3}{4}\right)^2 + \frac{1}{48} = s^2 - \frac{3}{2}s + \frac{7}{12}.$$

A direct integration also gives

$$\mathbb{E}[(W_r - s)^2 h(W_r)] = \frac{11}{12} - s + \left(-2s^2 + 4s - \frac{13}{6}\right)\log 2.$$

Subtracting the product of the first two moments yields

$$\mathrm{Cov}\big((W_r - s)^2, h(W_r)\big) = \frac{11}{12} - \log 2 - (1 - \log 2)s.$$

Therefore

$$\frac{\partial}{\partial(\beta^{-1})}\mathcal{U}_s(1/2, \beta)\Big|_{\beta=\infty} = (1 - \log 2)s - \left(\frac{11}{12} - \log 2\right).$$

The threshold s_β is the unique zero of this affine function. Hence, for every $s > s_\beta$, the derivative in the β^{-1} direction is positive at $\beta = \infty$, so for sufficiently small positive β^{-1}, equivalently for sufficiently large finite β, we have

$$\mathcal{U}_s(1/2, \beta) > \mathcal{U}_s(1/2, \infty).$$

□

The proposition identifies the point at which pure truncation becomes locally suboptimal, but it does not assign a single universal value of β. The optimal value depends on the oracle peak location s. To compute it, write $\alpha = 1/\beta$ and view $\mathcal{U}_s(1/2, \beta)$ as a one-dimensional function of $\alpha \geq 0$. Differentiating the tilted expectation gives

$$\frac{\partial}{\partial \alpha}\mathcal{U}_s(1/2, \beta) = \mathrm{Cov}_{f_{1/2,\beta}}\left(m_s(W), \log\left(\frac{W - \frac{1}{2}}{\frac{3}{2} - W}\right)\right).$$

Thus an interior optimum $\beta^\star(s)$ is obtained by solving this scalar covariance equation, while for $s \leq s_\beta$ the optimum remains the boundary value $\beta = \infty$, corresponding to pure truncation. Figure 5 plots the resulting finite optimizer for $s > s_\beta$: $\beta^\star(s)$ diverges as s approaches s_β from above, because the beneficial tilt is infinitesimal at the threshold, and decreases as s approaches one, where stronger concentration near the top proxy ranks becomes optimal. When the proxy is highly reliable $s \to 1$ and thus $\beta^\star \to 0$.

B Additional experiments and results

B.1 Raw rewards

Table 4: Raw reward for Llama 8B Tülu 3 SFT on UltraFeedback eval split (UF) and AlpacaEval (AE), with reporting token length.

	ArmoRM		Skywork-v2-Llama		Skywork-v2-Qwen		UF
Method	UF	AE	UF	AE	UF	AE	Len
Initial	$0.1300_{\pm 0.0008}$	$0.1370_{\pm 0.0003}$	$9.15_{\pm 0.18}$	$10.78_{\pm 0.05}$	$3.78_{\pm 0.15}$	$4.97_{\pm 0.03}$	$389.1_{\pm 8.0}$
DPO	$0.1497_{\pm 0.0002}$	$0.1620_{\pm 0.0003}$	$16.27_{\pm 0.10}$	$21.36_{\pm 0.18}$	$8.00_{\pm 0.07}$	$10.87_{\pm 0.07}$	$501.5_{\pm 2.3}$
REBEL	$0.1489_{\pm 0.0002}$	$0.1620_{\pm 0.0002}$	$16.35_{\pm 0.07}$	$22.15_{\pm 0.16}$	$8.00_{\pm 0.06}$	$11.20_{\pm 0.04}$	$516.1_{\pm 3.6}$
REBEL (rand)	$0.1436_{\pm 0.0002}$	$0.1568_{\pm 0.0003}$	$14.28_{\pm 0.10}$	$20.05_{\pm 0.09}$	$6.81_{\pm 0.02}$	$10.18_{\pm 0.03}$	$473.9_{\pm 5.2}$
QRPO	$0.1517_{\pm 0.0007}$	$0.1666_{\pm 0.0001}$	$17.41_{\pm 0.07}$	$21.08_{\pm 0.14}$	$8.24_{\pm 0.01}$	$10.31_{\pm 0.17}$	$730.1_{\pm 4.8}$
QRPO (rand)	$0.1465_{\pm 0.0001}$	$0.1610_{\pm 0.0001}$	$15.83_{\pm 0.09}$	$21.87_{\pm 0.12}$	$7.45_{\pm 0.06}$	$10.91_{\pm 0.06}$	$508.6_{\pm 2.8}$
BoNBoN	$0.1498_{\pm 0.0005}$	$0.1632_{\pm 0.0004}$	$18.30_{\pm 0.06}$	$24.29_{\pm 0.22}$	$8.89_{\pm 0.02}$	$12.25_{\pm 0.05}$	$548.3_{\pm 3.0}$
TUP (ours)	$0.1531_{\pm 0.0001}$	$0.1674_{\pm 0.0002}$	$\mathbf{20.20}_{\pm \mathbf{0.07}}$	$\mathbf{26.88}_{\pm \mathbf{0.09}}$	$\mathbf{9.75}_{\pm \mathbf{0.02}}$	$\mathbf{13.34}_{\pm \mathbf{0.02}}$	$645.5_{\pm 2.7}$

Tables 4 and 5 report the corresponding raw reward scores and generated lengths. On UltraFeedback, TUP remains strongest under both independent Skywork reward models, including on AlpacaEval. On Magpie Air, TUP remains competitive with the strongest baselines and improves over QRPO and BoNBoN under the independent Skywork evaluators. These raw results should be read together with the LC tables because several methods, including TUP and QRPO, tend to produce longer responses.

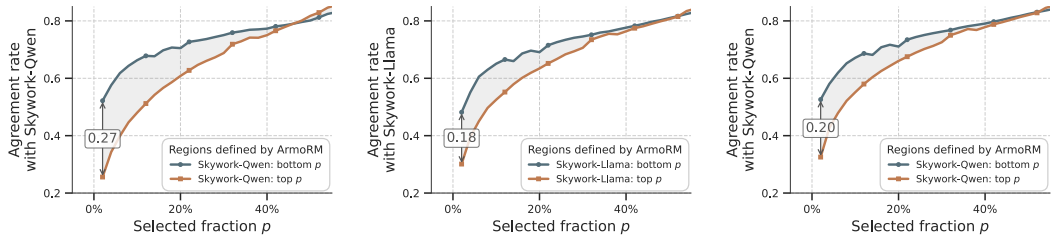

Figure 6: Additional cross-model agreement rates between ArmoRM and independent Skywork reward models. For each selected fraction p, the plots show the fraction of completions placed by ArmoRM in the top or bottom p region that the Skywork evaluator also places in the corresponding region. **Left:** UltraFeedback with Skywork-v2-Qwen. **Center:** Magpie Air with Skywork-v2-Llama. **Right:** Magpie Air with Skywork-v2-Qwen.

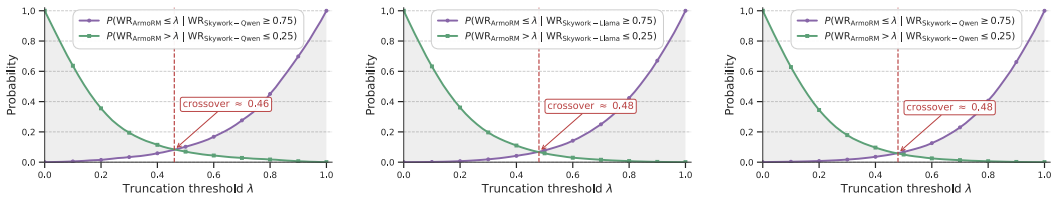

Figure 7: Additional cost–benefit curves for λ-truncation. Each plot compares the probability that truncation discards a completion ranked in the top 25% by the independent Skywork evaluator with the probability that it retains a completion ranked in the bottom 25% by that evaluator. **Left:** UltraFeedback with Skywork-v2-Qwen. **Center:** Magpie Air with Skywork-v2-Llama. **Right:** Magpie Air with Skywork-v2-Qwen.

Figures 6 and 7 extend the motivating observation in Figure 2 beyond the UltraFeedback/Skywork-v2-Llama setting. Across the additional dataset–evaluator combinations, ArmoRM-defined bottom regions are more consistently recovered by the independent Skywork evaluator than ArmoRM-defined top regions. The corresponding λ curves show the same truncation tradeoff: increasing λ reduces the chance of retaining completions that the independent evaluator ranks poorly, but also increases the chance of discarding completions it ranks highly. This supports using moderate truncation rather than an aggressive threshold.

Table 5: Raw reward for Llama 8B Tülu 3 SFT on Magpie Air eval split (MA) and AlpacaEval (AE), with reporting token length.

	ArmoRM		Skywork-v2-Llama		Skywork-v2-Qwen		MA
Method	MA	AE	MA	AE	MA	AE	Len
Initial	$0.1530_{\pm 0.0010}$	$0.1370_{\pm 0.0003}$	$10.01_{\pm 0.36}$	$10.78_{\pm 0.05}$	$4.99_{\pm 0.29}$	$4.97_{\pm 0.03}$	$331.6_{\pm 2.4}$
DPO	$0.1800_{\pm 0.0003}$	$0.1581_{\pm 0.0006}$	$19.23_{\pm 0.12}$	$18.82_{\pm 0.19}$	$11.93_{\pm 0.09}$	$9.69_{\pm 0.08}$	$484.5_{\pm 3.2}$
REBEL	$0.1795_{\pm 0.0003}$	$0.1591_{\pm 0.0007}$	$20.69_{\pm 0.31}$	$20.65_{\pm 0.19}$	$12.31_{\pm 0.14}$	$10.37_{\pm 0.12}$	$543.2_{\pm 4.5}$
REBEL (random)	$0.1778_{\pm 0.0002}$	$0.1565_{\pm 0.0003}$	$17.60_{\pm 0.20}$	$17.81_{\pm 0.14}$	$10.88_{\pm 0.14}$	$9.14_{\pm 0.08}$	$479.8_{\pm 4.4}$
QRPO	$0.1772_{\pm 0.0003}$	$0.1578_{\pm 0.0000}$	$18.23_{\pm 0.23}$	$18.99_{\pm 0.09}$	$10.25_{\pm 0.13}$	$9.30_{\pm 0.01}$	$554.0_{\pm 2.2}$
QRPO (random)	$0.1717_{\pm 0.0004}$	$0.1549_{\pm 0.0007}$	$16.46_{\pm 0.15}$	$18.20_{\pm 0.26}$	$9.28_{\pm 0.11}$	$9.03_{\pm 0.15}$	$473.2_{\pm 0.6}$
BoNBoN	$0.1654_{\pm 0.0003}$	$0.1453_{\pm 0.0008}$	$14.80_{\pm 0.21}$	$14.69_{\pm 0.12}$	$8.30_{\pm 0.06}$	$7.21_{\pm 0.13}$	$416.2_{\pm 1.6}$
TUP (ours)	$0.1791_{\pm 0.0004}$	$0.1583_{\pm 0.0006}$	$22.32_{\pm 0.23}$	$21.48_{\pm 0.25}$	$13.12_{\pm 0.12}$	$10.52_{\pm 0.03}$	$700.9_{\pm 4.3}$

Table 6: Raw reward for Mistral 7B SFT on Magpie Air eval split (MA) and AlpacaEval (AE), with reporting token length.

	ArmoRM		Skywork-v2-Llama		Skywork-v2-Qwen		MA
Method	MA	AE	MA	AE	MA	AE	Len
Initial	$0.1628_{\pm 0.0002}$	$0.1478_{\pm 0.0002}$	$15.99_{\pm 0.15}$	$16.55_{\pm 0.20}$	$9.86_{\pm 0.08}$	$8.77_{\pm 0.09}$	$493.1_{\pm 4.1}$
DPO	$0.1743_{\pm 0.0003}$	$0.1530_{\pm 0.0005}$	$22.71_{\pm 0.28}$	$20.64_{\pm 0.23}$	$16.75_{\pm 0.19}$	$11.84_{\pm 0.13}$	$750.1_{\pm 2.7}$
REBEL	$0.1593_{\pm 0.0004}$	$0.1424_{\pm 0.0014}$	$14.27_{\pm 0.20}$	$11.68_{\pm 0.20}$	$10.18_{\pm 0.10}$	$6.50_{\pm 0.13}$	$1375.7_{\pm 6.8}$
REBEL (random)	$0.1717_{\pm 0.0001}$	$0.1539_{\pm 0.0002}$	$\mathbf{23.29_{\pm 0.05}}$	$\mathbf{20.97_{\pm 0.04}}$	$\mathbf{17.19_{\pm 0.10}}$	$\mathbf{12.08_{\pm 0.01}}$	$861.0_{\pm 6.5}$
QRPO	$0.1710_{\pm 0.0006}$	$0.1532_{\pm 0.0002}$	$21.14_{\pm 0.13}$	$19.80_{\pm 0.06}$	$14.28_{\pm 0.03}$	$10.86_{\pm 0.11}$	$629.1_{\pm 2.2}$
QRPO (random)	$0.1522_{\pm 0.0005}$	$0.1301_{\pm 0.0005}$	$18.39_{\pm 0.12}$	$15.29_{\pm 0.13}$	$12.93_{\pm 0.03}$	$8.27_{\pm 0.11}$	$1155.9_{\pm 6.2}$
BoNBoN	$0.1655_{\pm 0.0003}$	$0.1491_{\pm 0.0006}$	$17.33_{\pm 0.26}$	$17.42_{\pm 0.13}$	$11.07_{\pm 0.14}$	$9.32_{\pm 0.09}$	$516.4_{\pm 1.7}$
TUP mild	$0.1757_{\pm 0.0004}$	$0.1537_{\pm 0.0004}$	$22.38_{\pm 0.03}$	$19.94_{\pm 0.10}$	$16.02_{\pm 0.01}$	$11.09_{\pm 0.05}$	$808.2_{\pm 10.1}$
TUP mid.	$\mathbf{0.1767_{\pm 0.0003}}$	$0.1562_{\pm 0.0002}$	$22.21_{\pm 0.18}$	$20.23_{\pm 0.21}$	$16.45_{\pm 0.14}$	$11.52_{\pm 0.07}$	$731.2_{\pm 3.9}$
TUP aggressive	$0.1749_{\pm 0.0005}$	$\mathbf{0.1564_{\pm 0.0006}}$	$20.47_{\pm 0.07}$	$19.33_{\pm 0.16}$	$15.31_{\pm 0.05}$	$10.91_{\pm 0.07}$	$744.4_{\pm 1.4}$

We also estimate the effective sharpness learned by the trained TUP checkpoint by reversing the construction used during training. During training, the truncated win-rate label w_λ determines the target policy-reference log-ratio through the configured sharpness β. After training, we instead observe the checkpoint log-ratio

$$\Delta_\theta(y \mid x) = \log \pi_\theta(y \mid x) - \log \pi_{\mathrm{ref}}(y \mid x),$$

and ask which value of β best explains its dependence on the reward-derived truncated win-rate. On the `test` split, we construct w_λ by ranking each response reward against the 12 reference completions for the same prompt and applying the same threshold $\lambda = 0.5$ used in training. We then fit the linear relation between $\Delta_\theta(y \mid x)$ and $z = \mathrm{logit}(w_\lambda)$ on the finite, unclipped subset. This gives an estimate of $\beta_{\mathrm{eff}} \approx 0.0147$. Thus, although training used the nominal value $\beta = 0.01$, the learned policy behaves on this diagnostic as if its effective sharpness is roughly 1.5 times larger.

B.2 Length-matched rewards

Table 7: TUP (with $\lambda = 0.5$) win rates against each benchmark method on similar-length AlpacaEval responses for Llama 8B Tülu 3 SFT trained on Magpie Air. For each comparison, we retain response pairs whose lengths differ by at most 10%. Values above 50% favor TUP and are shown in bold. The final column gives the number of retained pairs out of $2{,}415$.

Method	ArmoRM	Skywork-Llama	Skywork-Qwen	Prompts retained (of 2415)
DPO	**53.1%**	**55.3%**	49.1%	276
REBEL	**50.9%**	**50.2%**	45.8%	463
REBEL (random)	**64.9%**	**63.2%**	**61.8%**	238
QRPO	**51.4%**	**56.6%**	**56.6%**	415
QRPO (random)	**54.2%**	**53.1%**	46.7%	240
BoNBoN	**71.0%**	**60.7%**	**56.5%**	169
Average	**57.6%**	**56.5%**	**52.7%**	N/A

While the LC-reward results in Tables 1 and 2 already account for response length at the metric level, we seek further evidence that verbosity is not the sole contributor to TUP's strong performance. To

this end, Table 7 compares TUP with each baseline on responses of similar length. TUP is preferred over all benchmark methods under ArmoRM and Skywork-Llama. The results under Skywork-Qwen are more mixed across methods, although TUP's average win rate remains above 50% under all three reward models. Importantly, the matching criterion retains only a fraction of the response pairs and produces a different subset for each baseline. These results should therefore be viewed as complementary evidence rather than as an estimate of performance on the full test set. Together with the LC-reward results, they further suggest that TUP's strong performance is not solely due to its longer responses.

B.3 Global vs. per-prompt truncation

To estimate the potential headroom of prompt-adaptive truncation over a fixed global threshold λ, we consider an idealized empirical setting in which Skywork-Llama serves as the "oracle" evaluator. For each UltraFeedback prompt i, let $(y_i^{(1)}, y_i^{(2)}, \ldots, y_i^{(N)})$ denote the responses ranked by ArmoRM. We then:

1. Compute the ideal prompt-specific threshold
$$\lambda_i^* = \frac{\arg\max_j r_{\text{Skywork}}(y_i^{(j)}) - 1}{N},$$
which discards all responses ranked below the Skywork-best response by ArmoRM while retaining that response.
2. Select the single global threshold λ^* that maximizes the average Skywork-Llama win rate across all prompts.

To isolate the effect of the truncation threshold, we apply no within-tail reweighting. Under this idealized setting, the prompt-specific thresholds achieve an empirical Skywork-based win rate of 0.9325, compared with 0.8820 for the best fixed global threshold. Thus, although prompt-specific truncation offers some additional headroom, the fixed global threshold captures most of the attainable win rate in this analysis.

C Implementation details

C.1 Codebase, models, and data

The experiments in Tables 1–6 use the released QRPO benchmark data rather than regenerating reference completions or reward annotations. These data contain prompt-specific reference-completion pools scored by ArmoRM.

All experiments are implemented using the official QRPO repository[2], together with its released preprocessed reference datasets[3]. We use Llama 8B Tülu 3 SFT as the initial policy. On Mistral 7B we run a dedicated supervised fine-tuning (STF) on the original model (Mistral-7B-Instruct-v0.2), we closely followed QRPO code, using the prompt–chosen-response from Magpie-Air-DPO-100K-v0.1. After filtering sequences to at most 2,048 tokens, the training set contained 97,812 examples, with 1,992 held out for evaluation. We trained for one epoch in `bfloat16` using an effective batch size of 128 across four GPUs, a learning rate of 5×10^{-7}, 10% warm-up, and cosine decay, without LoRA or other parameter-efficient tuning methods. To reproduce, please refer to our or QRPO GitHub repository.

The training reward model in the released data is `RLHFlow/ArmoRM-Llama3-8B-v0.1`. For transfer evaluation, we rescore generated completions with `Skywork/Skywork-Reward-V2-Llama-3.1-8B` and
`Skywork/Skywork-Reward-V2-Qwen3-8B`.

For AlpacaEval, we report the win rate and length-controlled win rate against the default `gpt4-turbo` reference outputs, which correspond to `gpt-4-1106-preview`; `gpt-4o` is used only as the judge due to the deprecation of `gpt-4-1106-preview`.

The UltraFeedback setting contains $61{,}024$ training prompts and a held-out set of $1{,}995$ prompts, split into 998 checkpoint-selection prompts and 997 reward-reporting prompts. The Magpie Air setting

[2] `https://github.com/CLAIRE-Labo/quantile-reward-policy-optimization.git`
[3] `https://huggingface.co/collections/skandermoalla/qrpo-reference-datasets`

Table 8: RewardBench standings (regardless of public availability) for the reward and judge models used in our experiments, retrieved in April 2026. Global Rank denotes the model's overall position on the RewardBench leaderboard, and Type Rank denotes its position within the corresponding model category. The asterisk marks `gpt-4o`; because AlpacaEval uses an API-hosted judge, the exact served model version may not match the RewardBench entry.

Model	Role in this work	Model type	Global Rank	Type Rank
ArmoRM-Llama3-8B-v0.1	Training proxy reward model	Classifier	#34	#26
Skywork-Reward-V2-Llama-3.1-8B	Evaluation reward model	Classifier	#1	#1
Skywork-Reward-V2-Qwen3-8B	Evaluation reward model	Classifier	#6	#3
`gpt-4o`*	GPT judge	Generative	#36	#10

contains $97{,}812$ training prompts and a held-out set of $1{,}992$ prompts, split into 996 checkpoint-selection prompts and 996 reward-reporting prompts. AlpacaEval contains 805 evaluation prompts. The QRPO benchmark reference pools contain 50 reference completions per prompt.

For DPO, REBEL, and QRPO, we use the best-performing non-SFT, off-policy configurations reported in Tables 12 and 14 of Matrenok et al. [26]. We use the `offpolicy2best` and `offpolicy2random` variants as reported in Tables 9 and 10. BoNBoN and TUP are not included in the original QRPO benchmark, so we perform a separate hyperparameter search for these methods.

C.2 Reward models used for evaluation

Table 8 summarizes the external reward models used in our experiments. ArmoRM is used only as the offline proxy reward model for training labels, whereas the Skywork-v2 models and `gpt-4o` are used only for evaluation. The RewardBench ranks are provided only as external context, since leaderboard positions can change as new models are added.

C.3 TUP implementation

TUP uses the same precomputed completion pools and ArmoRM scores as the QRPO benchmark. For each prompt, we convert the ArmoRM scores in the pool into empirical in-pool win-rates, apply the shifted truncation $\hat{w}_{\lambda,r} = \max(\hat{w}_r - \lambda, 0)$, and train with the BCE objective in Algorithm 1. Relative to the QRPO pipeline, the implementation only changes the scalar training label, the known intercept, and the loss used. In the reported experiments the intercept uses the population normalizer $\beta \log Z_{\lambda,\beta}$; Appendix A.1 also gives the exact finite-pool alternative $Z_{K,\lambda,\beta}$ for the empirical-rank convention used in Algorithm 1.

Numerical evaluation of the population normalizer. Although Proposition 3.1 writes $Z_{\lambda,\beta}$ as an incomplete Beta function, the implementation evaluates the equivalent continuous integral directly with high-precision arithmetic. We do not use standard regularized incomplete-Beta routines such as `scipy.special.betainc`, since these APIs normalize by the complete Beta function and assume positive shape parameters, whereas here the second parameter is $1 - 1/\beta < 0$ for the β values used in our sweeps. We therefore compute

$$Z_{\lambda,\beta} = \int_0^{1-\lambda} u^{1/\beta}(1-u)^{-1/\beta}\, du$$

using `mpmath` with `mp.workdps(256)`. This avoids the underflow that can occur in standard double-precision quadrature for small β and makes the BCE intercept $\beta \log Z_{\lambda,\beta}$ reproducible.

C.4 Hyperparameter search and model selection

For BoNBoN, we use only the IPO-BoN component of the original objective. This gives the closest functional comparison in our setting, which is offline BoN-style distillation without an additional SFT loss. We tune only the learning rate:

$$\text{lr} \in \{1\text{e}{-6}, 3\text{e}{-7}, 1\text{e}{-7}\}.$$

For TUP using Llama model on UltraFeedback and Magpie Air, we tune the learning rate, preference sharpness, and truncation threshold over

$$\text{lr} \in \{1\text{e}{-6}, 3\text{e}{-7}, 1\text{e}{-7}\}, \quad \beta \in \{3\text{e}{-3}, 1\text{e}{-2}, 3\text{e}{-2}\}, \quad \lambda \in \{0.2, 0.5, 0.8\}.$$

Table 9: Training hyperparameters Llama 8B Tülu 3 SFT on UltraFeedback reported in Table 1, Table 2 and Table 4.

Method	Dataset variant	β	lr	Num ref	λ	Checkpoint
DPO	offpolicy2best	0.01	3e-07	6	-	300
REBEL	offpolicy2best	1e-06	3e-07	6	-	200
REBEL (rand.)	offpolicy2random	1e-06	1e-06	6	-	100
QRPO	offpolicy2best	3e-04	3e-07	3	-	100
QRPO (rand.)	offpolicy2random	3e-04	3e-07	3	-	100
BoNBoN	offpolicy2best	1e-03	1e-06	6	-	476
TUP mild	offpolicy2random	0.01	1e-07	6	0.2	400
TUP mid.	offpolicy2random	0.01	1e-07	6	0.5	400
TUP aggressive	offpolicy2random	0.01	1e-07	6	0.8	476

Table 10: Training hyperparameters Llama 8B Tülu 3 SFT on Magpie Air reported in Table 1, Table 2 and Table 5.

Method	Dataset variant	β	lr	Num ref	λ	Checkpoint
DPO	offpolicy2random	0.01	1e-06	6	-	480
REBEL	offpolicy2best	1e-06	1e-07	6	-	640
REBEL (rand.)	offpolicy2random	1e-04	1e-06	6	-	320
QRPO	offpolicy2best	1e-03	1e-07	1	-	320
QRPO (rand.)	offpolicy2random	1e-03	1e-07	1	-	320
BoNBoN	offpolicy2best	1e-03	1e-06	6	-	764
TUP mild	offpolicy2random	0.01	1e-07	6	0.2	320
TUP mid.	offpolicy2random	0.01	1e-07	6	0.5	320
TUP aggressive	offpolicy2random	0.01	1e-07	6	0.8	320

For Mistral 7B on Magpie Air, due to computational constraints, we ran the search on a narrower set of hyper parameters:

$$\text{lr} \in \{1\text{e}{-6}, 3\text{e}{-7}\}, \quad \beta \in \{1\text{e}{-2}, 3\text{e}{-2}\}, \quad \lambda \in \{0.2, 0.5, 0.8\}.$$

Checkpoint selection uses only the validation split. We select checkpoints by length-controlled reward under ArmoRM. For both model families, UltraFeedback checkpoints are saved and evaluated every 100 steps and Magpie Air checkpoints are saved and evaluated every 160 steps. For UltraFeedback, we only consider checkpoints with mean generated length below 650 tokens. For Magpie Air, we do not impose an analogous length threshold. Tables 9 and 10 report the selected hyperparameters and checkpoints for each method.

C.5 Training settings

Unless stated otherwise, training settings follow the QRPO benchmark configuration. Training uses `bfloat16` precision with Accelerate and DeepSpeed ZeRO-1. We use a cosine learning-rate schedule with warmup ratio 0.1. Gradient clipping is effectively disabled in the Llama benchmark runs by setting `max_grad_norm` to 10^8.

The per-device training batch size is 2, the per-device evaluation batch size is 4, and gradient accumulation is 16. With 4 GPUs, this gives an effective training batch size of 128.

The maximum prompt length, maximum completion length, and maximum total sequence length are all 2048. The released reference completions were generated with temperature 1.0 and top-$p = 1.0$. For online evaluation, we generate one completion per prompt using temperature 0.6, top-$p = 0.9$, and a maximum of 2048 new tokens.

C.6 Evaluation and uncertainty

For reward-model evaluation, we follow the QRPO evaluation pipeline. For each generated completion, the evaluator computes both raw reward and length-controlled reward using the reference-completion pool for the same prompt. For UltraFeedback and Magpie Air, we use the released QRPO reference pools. For AlpacaEval, which is not part of the released QRPO reference data, we construct the reference pool by generating 16 completions per prompt from the base model using vLLM. To compute length-controlled metrics under a given reward model, we score both the generated completions and the corresponding prompt-specific reference pools with that reward model. The length-controlled score normalizes reward and length relative to the prompt-specific reference pool,

Table 11: Training hyperparameters Mistral 7B SFT on Magpie Air reported in Table 3 and Table 6.

Method	Dataset variant	β	lr	Num ref	λ	Checkpoint
DPO	offpolicy2random	0.03	3e-07	6	-	480
REBEL	offpolicy2best	1e-04	1e-07	6	-	480
REBEL (rand.)	offpolicy2random	1e-04	3e-07	6	-	320
QRPO	offpolicy2best	3e-03	1e-07	1	-	764
QRPO (rand.)	offpolicy2random	3e-03	1e-06	1	-	764
BoNBoN	offpolicy2best	1e-03	1e-07	6	-	480
TUP mild	offpolicy2random	0.01	1e-06	6	0.2	764
TUP mid.	offpolicy2random	0.01	1e-06	6	0.5	764
TUP aggressive	offpolicy2random	0.01	1e-06	6	0.8	764

then removes the estimated linear effect of length. Prompts with degenerate reference-length variance are masked by the evaluation code.

For reward-model metrics, we report uncertainty as the sample standard error across three generation seeds. For AlpacaEval, we replace the recently `gpt-4-1106-preview` judge used in earlier benchmark configurations with `gpt-4o`. This change may make the AlpacaEval scores not directly comparable to results computed with the older judge.

C.7 Deviation from the QRPO implementation

The QRPO codebase constructs training examples using both chosen and rejected completions by default. For TUP, we instead use a single randomly selected response per prompt. This matches the scalar-label formulation of Algorithm 1, where each completion receives one shifted-truncated win-rate label. It also means that TUP receives less per-prompt training signal than methods that use both responses.

C.8 Compute resources

The reported training runs were performed on a machine with 4 H200 GPUs and 64 CPU cores.

C.9 Compliance and asset licenses

Table 12 summarizes the external code, model, API, and dataset assets used in the experiments. License and terms information was retrieved from the official repository, model cards, dataset cards, and provider documentation on April 30, 2026. We use these assets only for offline training or evaluation as described in Appendix C, and we do not redistribute the underlying external model checkpoints, API-hosted judge, or source datasets as part of this submission.

Table 12: External assets used in the experiments and their reported licenses or terms. For the QRPO reference data, the table lists the released preprocessed datasets actually loaded by our experiments; these are derived reference-completion pools and rewards rather than newly collected data in this work.

Asset	Role in this work	Reported license or terms
`QRPO reference codebase`	Experimental framework and baseline implementation.	MIT license.
QRPO UltraFeedback reference datasets `offpolicy2best`, `offpolicy2random`	Preprocessed UltraFeedback reference-completion pools and ArmoRM scores used for training and evaluation.	MIT license.
QRPO Magpie Air reference datasets `offpolicy2best`, `offpolicy2random`	Preprocessed Magpie Air reference-completion pools and ArmoRM scores used for training and evaluation.	MIT license for the QRPO preprocessed releases; the upstream Magpie Air dataset card does not declare a separate license field.
`openbmb/UltraFeedback`	Source instruction-tuning dataset underlying the UltraFeedback QRPO setting.	MIT license.
`tatsu-lab/alpaca_eval`	AlpacaEval prompt set used for GPT-judge evaluation.	CC-BY-NC-4.0 license.
`allenai/reward-bench`	RewardBench leaderboard context for the reward and judge models in Table 8.	ODC-BY-1.0 license.
`allenai/Llama-3.1-Tulu-3-8B-SFT`	Initial policy for the reported Llama benchmark runs.	Llama 3.1 Community License.
`mistralai/Mistral-7B-Instruct-v0.2`	Initial policy for the reported Mistral benchmark runs (SFT on this model was ran separately).	Mistral Community License.
`RLHFlow/ArmoRM-Llama3-8B-v0.1`	Offline proxy reward model used in the released QRPO labels and for checkpoint selection.	Llama 3 Community License.
`Skywork/Skywork-Reward-V2-Llama-3.1-8B`	Independent reward model used for transfer evaluation.	Llama 3.1 Community License.
`Skywork/Skywork-Reward-V2-Qwen3-8B`	Independent reward model used for transfer evaluation.	Apache-2.0 license.
`gpt-4o`	API-hosted judge for AlpacaEval.	Proprietary OpenAI API model used under OpenAI service terms; no weights are redistributed.

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 Section 1 state the paper's main claims: TUP as a truncate-bad, upweight-good policy, its offline BCE training procedure, its theoretical motivation, and its empirical evaluation. These claims are supported by the method in Sections 3.1 and 3.3, the theory in Section 3.2, and the experiments in Section 4.

   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: Section 5 discusses the main practical limitation, namely that TUP has two hyperparameters, λ and β, that must be tuned on validation performance, as well as further limitations and future work (exposure to reward hacking, and the use of fixed global rather than prompt-specific λ and β). Section 3.2 notes that the theory is idealized and does not directly explain how a trained model benefits from it.

   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: The main-text results (Theorem 3.2 and Proposition 3.3) are stated informally in Section 3.2, together with their key assumptions. The formal statements, full assumptions, and complete proofs are in Appendix A, in particular Appendices A.2.1 and A.2.2 (Theorem A.3 and Proposition A.5).

   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 4 describes the experimental setup and main results, Algorithm 1 specifies the training procedure, and Appendix C gives the codebase, data and splits, models, hyperparameter search, training settings, evaluation protocol, and the deviation from the QRPO implementation needed to reproduce the reported experiments. The TUP code is also publicly available (see the repository linked in Section 1).

   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: [Yes]

   Justification: The TUP implementation and experiment scripts are publicly available at github.com/yarinbar/truncate-bad-upweight-good (linked in Section 1) under an MIT license, with a README describing the setup and the commands used to run the experiments. The experiments build on the public QRPO repository and its released preprocessed reference datasets, as described in Appendix C.

   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: The experimental setup is described in Section 4, including datasets, evaluators, baselines, and how baseline and TUP configurations are chosen. Appendix C provides the implementation details, data splits, model and data sources, hyperparameter search, training settings, and checkpoint selection, with selected settings reported in Tables 9, 10 and 11.

   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: The main results (Tables 1, 2 and 3), the raw-reward tables in Appendix B.1, and the λ–β ablation (Figure 4) report standard errors. For the reward-model metrics, these are the sample standard error across three generation seeds (Appendix C); the ArmoRM errors in the main tables are negligible (fourth decimal place) and are omitted there for legibility (Section 4).

   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 C (Compute resources) states that the reported training runs were performed on a machine with 4 H200 GPUs and 64 CPU cores, and the same appendix gives the per-device and effective batch sizes.

   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 is an offline methods and evaluation study using released QRPO benchmark data, preprocessed reference datasets, and publicly identified model and reward-model resources, as described in Section 4 and Appendix C, whose licenses and terms of use are listed in Appendix C.9. We do not identify any aspect of the reported research that requires deviation from the NeurIPS Code of Ethics.

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: Section 5 discusses a potential positive impact (TUP may support safety alignment when unsafe or otherwise undesirable completions are consistently ranked low by the proxy reward, so that truncation removes them from the target support) and notes that TUP remains exposed to reward hacking. TUP is a general-purpose offline alignment method evaluated on existing public data, and we do not identify a direct path to specific negative applications beyond those common to reward-model-based alignment in general.

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 introduces a training method and evaluates it using existing released QRPO reference data and existing models; it does not describe releasing new high-risk data or model artifacts requiring safeguards. The relevant experimental assets and implementation setup are documented in Appendix C.

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: [Yes]

    Justification: Section 4 and Appendix C credit the QRPO framework, released reference datasets, initial policy, reward models, and GPT judge, with URLs for the QRPO repository and datasets. Appendix C.9 and Table 12 list the license or terms of use of each external code, model, API, and dataset asset.

    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: [Yes]

    Justification: The new asset of this paper is the TUP implementation, released at the repository linked in Section 1 with a README documenting the setup, training, and evaluation, and an MIT license. The paper introduces no new datasets; the experiments use the released QRPO reference data (Appendix C).

    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: [N/A]

    Justification: The paper does not report crowdsourcing experiments or research with human subjects; it evaluates methods using released QRPO benchmark data, existing model and reward-model resources, and offline evaluation protocols described in Section 4 and Appendix C.

    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: [N/A]

    Justification: Section 4 and Appendix C describe an offline methods and evaluation study using released QRPO benchmark data, existing models, reward models, and GPT-judge evaluation, with no crowdsourcing or human-subject study. Therefore IRB approval and participant-risk disclosure are not applicable.

    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: [N/A]

    Justification: The core methodological contribution is the TUP rank-based target policy and BCE training objective in Sections 3.1 and 3.3; the LLM policy, reward models, and `gpt-4o` judge are standard experimental assets or evaluators rather than important, original, or non-standard LLM components of the method, and are described in Section 4 and Appendix C.

    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.