PREFERENCE STEALING FROM BEST-OF-N: THE PRIVACY RISK OF INFERENCE-TIME ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Inference-time alignment methods are gaining popularity since they improve the helpfulness and safety of language models' responses without training. Most methods rely on a reward model (RM) to raise the probability of good responses. The RM encodes preferences from expensive, proprietary, and often sensitive human-annotated training data. Both the RM itself and these preferences are therefore valuable targets for attackers. However, prior work on preference privacy has focused on training time, leaving the risk at inference time unexplored. We first raise the concern that inference-time alignment methods might leak the RM's preferences through the responses they serve. We take Best-of-N (BoN) sampling, one of the most widely used methods, as the representative case. BoN keeps its RM in the serving loop, so every served response implicitly reveals a preference signal from the RM. We study how attackers might acquire the RM's preferences under different deployment scenarios, which expose different amounts of information about BoN's alignment process. Our experiments show that even under the strictest scenario in which BoN serves only the final response, attackers can still recover the preferences behind the RM's training data with about 75% accuracy, against 50% for chance. Finally, we take a first step toward a defense through output perturbation, yet the attack survives almost intact. BoN keeps leaking the RM's preferences, and closing the leak remains an open problem.

1 INTRODUCTION

Aligning a large language model (LLM) no longer ends at training. A growing family of inference-time alignment methods steers a frozen LLM policy at inference time, trading extra computation for more helpful and safer responses. This trade becomes increasingly attractive as inference-time compute scales (Welleck et al., 2024; Zhang et al., 2025). Best-of-N (BoN) sampling is the simplest and most widely used among these methods. Given a prompt, BoN draws N candidate responses from the base policy π_{ref}, scores each with a reward model (RM) $r^\star$, and returns the highest-scoring one. BoN needs no policy updates, wraps any black-box generator, and asymptotically attains the optimal trade-off between expected reward and KL divergence from π_{ref} (Yang et al., 2024). It already underlies systems from WebGPT's best-of-64 (Nakano et al., 2021) to verifier-reranked reasoning (Cobbe et al., 2021), and it is the standard baseline in inference-time scaling studies (Snell et al., 2025; Brown et al., 2024).

A key component that needs to be kept private is the reward model, which provides steering signals in inference-time alignment methods. The RM learns human preferences from expensive, proprietary, and often sensitive human-annotated datasets. For example, InstructGPT's RM learned from prompts that users submitted to the OpenAI API (Ouyang et al., 2022). Open preference data such as the Chatbot Arena votes come from in-the-wild user conversations (Zheng et al., 2023), and LMSYS-Chat-1M releases a million such conversations (Zheng et al., 2024). All of these preferences are encoded in the RM. If the RM leaks them, two parties will be harmed. **(i) Users and annotators**, whose preferences are memorized by the RM. These preferences are often sensitive and collected from conversations that carry personal disclosures, from relationships and political or religious views to detailed sexual preferences and drug use habits (Mireshghallah et al., 2024). **(ii) The RM's owner**. Training a commercial-grade RM is expensive. For example, Meta collected about 1.4M human comparisons for Llama 2's reward models (Touvron et al., 2023). The

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

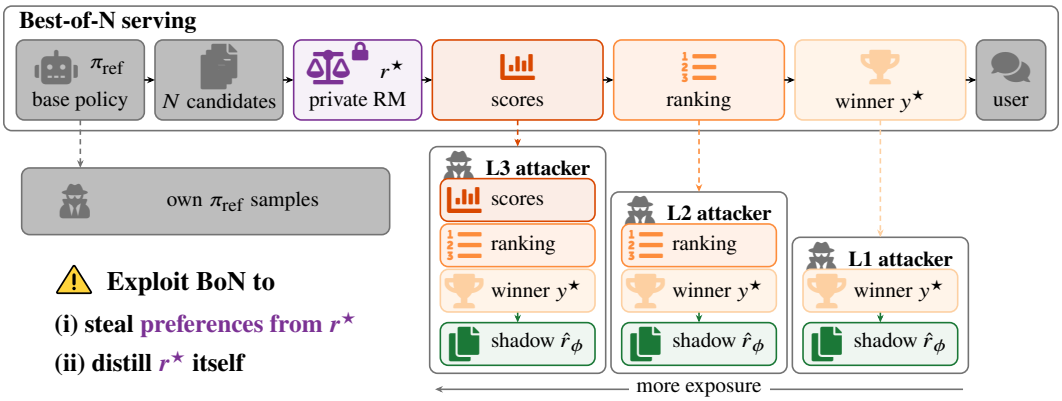

Figure 1: **Best-of-N serving leaks the preferences inside its private reward model.** A BoN deployment draws N candidates from the public base policy π_{ref}, scores them with the private RM $r^\star$, and returns the winner $y^\star$. An attacker sees only what the deployment returns. At L1 it sees the winner only, at L2 also the ranked candidates, and at L3 also the reward scores. Attacker trains a shadow model from these observations and samples of π_{ref} $\hat{r}_\phi$ without ever querying $r^\star$.

trained RM is therefore valuable intellectual property and a natural target for model-extraction attacks (Tramèr et al., 2016; Krishna et al., 2020; Carlini et al., 2024), in which an adversary distills the RM's behavior for its own use. The adversary can then benefit from the distilled RM through multiple approaches, for instance reinforcement learning from AI feedback (RLAIF), where the distilled RM replaces human labelers (Bai et al., 2022; Lee et al., 2024), or reward hacking, where the adversary searches offline for responses that $r^\star$ scores high but humans would not (Gao et al., 2023).

BoN is where this risk becomes concrete at inference time, and to our knowledge, we are the first to show that it acts as an exploitable side channel for adversaries. BoN keeps its RM in the serving loop beyond the training phase. Every served response is the one that $r^\star$ scored highest among all N candidates, so it carries a preference signal that an adversary was never meant to see. However, most prior work on preference privacy studies the training-time setting, for instance through membership inference on preference data (Feng et al., 2025) and on post-training prompts (Liu et al., 2025b), and differentially private reward learning (Ray Chowdhury et al., 2024; Wu et al., 2024). These studies ask what the aligned policy or the RM reveals about its training data, whereas the risk we describe arises at inference time. To the best of our knowledge, such a privacy risk in BoN has not been identified in the literature, which raises an important research question: *Can an adversary leverage BoN as a side channel to recover the preferences behind $r^\star$'s training data, or even to distill $r^\star$ itself?*

Contributions.

1. **We identify the privacy risk in inference-time alignment.** Any RM-based inference-time alignment method that selects responses via a private reward model creates a side channel on the preferences that model encodes. Exploiting this loophole harms both the users and annotators whose preference data are used to train the RM, and also the RM's owner whose costly model is thereby distilled. We are the first to reveal this side channel, and we study BoN as the most widely deployed representative.
2. **We measure how severely BoN leaks under different deployment scenarios.** We formalize the attack across three nested levels of deployment exposure (§4) and measure it against five open-weight RMs. Even at the strictest level which serves only the BoN winner, a cheap shadow model[1] trained on those winners still recovers the preferences behind $r^\star$'s training data up to 75% accuracy, against 50% for chance (Table 1). Once the deployment also exposes the ranking, the shadow model agrees with $r^\star$ on at least 90% of pairs, including prompts far from $r^\star$'s training

[1]We use shadow model for the adversary's distilled copy of $r^\star$, not for the auxiliary models of membership inference (Shokri et al., 2017).

data (Table 2), so the adversary ends up with an approximate copy of $r^\star$ itself. The true leak may be worse as our attack is unoptimized.

3. **We make a first attempt at a defense.** We take an initial step toward defending against this leak by perturbing BoN's winner selection so that it reveals less about $r^\star$. We add noise to $r^\star$'s scores before the winner is chosen, following the output-perturbation defenses against model extraction (Orekondy et al., 2020; Kariyappa & Qureshi, 2020). This idea turns out to coincide with soft-BoN, which samples the winner from a softened distribution over the candidates and was originally proposed to keep BoN's output closer to the base policy (Verdun et al., 2025). Our finding is that these methods do carry a modest privacy benefit, but at the cost of BoN's utility. Even when the deployment gives up 27% of BoN's reward gain, the attack loses at most 7 points of accuracy (§5.4). Closing the leak without giving up the gain of BoN thus remains open, and we hope our results draw the community's attention to this direction.

2 Related work

Reward recovery and preference privacy. Preference or reward signals can be recovered from observed behavior. Inverse reinforcement learning (IRL) learns a reward function from human demonstrations (Ng & Russell, 2000; Ziebart et al., 2008), and recent IRL studies recover the human preferences behind an aligned LLM by comparing its responses before and after alignment (Joselowitz et al., 2025; Patel et al., 2025). These studies confirm the premise of our attack: an aligned policy carries information about the reward that aligned it. The recovery is indirect, however, because the reward acted on the policy once during training. In inference-time alignment, the RM never leaves the serving loop. Each served response is a choice the RM itself made among the candidates, and under BoN the likelihood of this choice has a closed form in the reward (§3). Our attack maximizes this likelihood directly (§4). On the training-time alignment side, the privacy risk of preference data is well studied. RL fine-tuning on human preferences is routine in LLM post-training (Ouyang et al., 2022), yet these preferences carry annotators' sensitive information. Membership inference attacks show that a fine-tuned LLM reveals whether a preference example (Feng et al., 2025) or a training prompt (Liu et al., 2025b) was in its fine-tuning data, and differentially private reward learning limits what an RM reveals about any single comparison (Ray Chowdhury et al., 2024; Wu et al., 2024). Both lines of work treat the trained model as the object to attack or protect. Neither considers what the RM's decisions reveal once it is deployed.

Model extraction. The RM's owner is the other harmed party. A model extraction attack trains a substitute model to imitate a target from the target's outputs (Jagielski et al., 2020; Zhao et al., 2025), and the form of the attack follows how much of the output the attacker sees. For instance, a classifier can be extracted from its predicted labels alone (Sanyal et al., 2022; Chen et al., 2024), and so can the guardrail of a commercial LLM (Yao et al., 2025). Text rankers and recommenders can be extracted from the order in which they return items, without any score (Liu et al., 2022; Yue et al., 2021). When the scores themselves are exposed, as with early prediction APIs, extraction becomes easier still (Tramèr et al., 2016). Our three exposure levels (§4) follow these three regimes. Most prior studies examine one regime on its own target, however, whereas we measure all three against the same RM. This lets us quantify what each additional piece of the RM's output costs its owner. Our BoN setting also tightens the strictest regime. In label-only extraction, the attacker picks the input and observes the label. In contrast, the adversary observes the chosen candidate but not the rejected ones at our winner-only level, so even the input to the RM's decision is hidden. The concurrent JudgeStealer extracts LLM evaluators, including an RM, from the scores they return (Chen et al., 2026), which requires querying the target under the most permissive regime above. Our attack never queries the RM directly and succeeds from the winner alone. We show that this suffices to distill an approximate copy of the RM, which is an underexplored privacy risk to its owner (§1).

3 Preliminaries

Best-of-N sampling. Given a prompt x, BoN draws N candidates $y_1, \ldots, y_N \sim \pi_{\text{ref}}(\cdot \mid x)$ from the base policy and returns the winner $y^\star = \arg\max_i r^\star(x, y_i)$, the candidate that the reward model $r^\star$ scores highest. Sessa et al. (2025) give the winner distribution π_{BoN} in closed form when no two

candidates tie:

$$\pi_{\mathrm{BoN}}(y^\star \mid x) = N\,\pi_{\mathrm{ref}}(y^\star \mid x)\,F_r(x,y^\star)^{N-1},\ \text{where } F_r(x,y^\star) = \Pr_{y'\sim\pi_{\mathrm{ref}}}\big[\,r(x,y') < r(x,y^\star)\,\big]. \tag{1}$$

$F_r(x,y^\star)$ is the probability that a π_{ref} sample scores below $y^\star$ under a reward model r. We take advantage of this distribution to construct the attacker's training objective in (§4).

Bradley–Terry preference loss. Usually, a reward model is trained on pairwise comparison data and uses Bradley–Terry loss to inject the human preferences among the data. (Stiennon et al., 2020; Ouyang et al., 2022; Liu et al., 2024):

$$\ell_{\mathrm{BT}}(r) \;=\; -\log\sigma\big(r(x,y_c) - r(x,y_r)\big), \tag{2}$$

where y_c is the chosen response, y_r the rejected one, and σ the logistic function. We show later (§4) that the attacker's training objective takes a closely related form, with π_{ref} samples in place of rejected responses.

Listwise ranking loss. Beyond a single comparison pair (y_c, y_r), when a list of responses and their ranking are available, the Plackett–Luce model (Luce, 1959; Plackett, 1975) is a better fit than a pairwise loss: its maximum likelihood estimator is asymptotically more efficient than splitting the ranking into pairs (Zhu et al., 2023), and a rank-weighted listwise objective outperforms pairwise ones on ranked preference data (Liu et al., 2025a). Writing the ranked list as $(y_{(1)}, y_{(2)}, \ldots, y_{(N)})$ from most to least preferred (i.e., $r(x,y_{(1)}) > r(x,y_{(2)}) > \cdots > r(x,y_{(N)})$), the ListMLE loss (Xia et al., 2008) fits the reward model to the whole ranking:

$$\ell_{\mathrm{LM}}(r) \;=\; -\sum_{k=1}^{N-1} \log \frac{\exp r(x,y_{(k)})}{\sum_{j=k}^{N}\exp r(x,y_{(j)})}. \tag{3}$$

In fact, the Bradley–Terry loss (2) is its $N{=}2$ case. We also use this ListMLE loss to construct an attack, as detailed in §4.

4 Exploiting the Leak: A Preference Distillation Attack

The attacker's goal is to train a shadow reward model $\hat{r}_\phi$ with parameters ϕ to reproduce the preferences of the true RM $r^\star$. Concretely, we assume the attacker (i) can sample from the public base policy π_{ref}, (ii) knows that the inference-time alignment method is BoN, and (iii) observes information exposed by the BoN-aligned policy π_{BoN}. The first two assumptions are independent of how BoN is deployed, whereas the third depends on the endpoint: different deployments may expose different amounts of information about the selection process. We therefore consider three nested exposure levels, illustrated in Fig. 1, where each higher level contains all information available at the preceding levels.

We first show how an attacker can exploit these observations to distill a shadow reward model, and then study defenses that perturb the BoN selection process to weaken the leaked preference signal.

4.1 Preference Distillation Attack

L1: winner only The most direct objective is maximum likelihood. Each winner $y^\star$ is a draw from π_{BoN}, and $r^\star$ is the only unknown in its closed-form expression (1). Replacing $r^\star$ with our shadow model $\hat{r}_\phi$ there and maximizing the log-likelihood of $y^\star$ over ϕ gives

$$\begin{aligned}&\max_\phi\ \log\big(N\,\pi_{\mathrm{ref}}(y^\star \mid x)\,F_\phi(x,y^\star)^{N-1}\big)\\ &\quad=\ \max_\phi\ \Big[\log N + \log\pi_{\mathrm{ref}}(y^\star \mid x) + (N{-}1)\log F_\phi(x,y^\star)\Big].\end{aligned} \tag{4}$$

Only the last term depends on ϕ, so the objective reduces to maximizing $F_\phi(x,y^\star)$, the probability that $\hat{r}_\phi$ scores $y^\star$ higher than a π_{ref} sample:

$$\max_\phi\ \log F_\phi(x,y^\star). \tag{5}$$

This objective matches what the attacker holds at this level: BoN winners $y^\star \sim \pi_{\text{BoN}}$, each selected by $r^\star$ from N candidates, and the attacker's own independent draws from π_{ref}, which pass through no selection. An intuitive way to capture $r^\star$'s preference is to make the shadow score $y^\star$ above those π_{ref} samples, and that is exactly what (5) does.

Next we show how to estimate $F_\phi(x, y^\star)$. Drawing K fresh references $\tilde{y}^{(1)}, \ldots, \tilde{y}^{(K)} \sim \pi_{\text{ref}}(\cdot \mid x)$ for prompt x gives the Monte-Carlo estimate $\frac{1}{K}\sum_k \mathbf{1}[\hat{r}_\phi(x, \tilde{y}^{(k)}) < \hat{r}_\phi(x, y^\star)]$. The indicator function is non-differentiable almost everywhere, so we relax it to a sigmoid of the score difference,

$$\widehat{F}_\phi(x, y^\star) \;=\; \frac{1}{K}\sum_{k=1}^{K} \sigma\big(\hat{r}_\phi(x, y^\star) - \hat{r}_\phi(x, \tilde{y}^{(k)})\big). \tag{6}$$

Substituting $\widehat{F}_\phi$ into (5) gives the per-prompt training loss of the attack (`cdf` in §5),

$$\ell_{\text{L1}} \;=\; -\log \max\big(\widehat{F}_\phi(x, y^\star), \epsilon\big), \tag{7}$$

where $\epsilon = 10^{-6}$ is a floor for numerical stability. Starting from the winner's likelihood, we eventually arrive at a loss that resembles the Bradley–Terry loss (2), with K fresh π_{ref} samples in place of the rejected response. Pukdee et al. (2026) show that Bradley–Terry training on exactly such pairs, a winner against a fresh π_{ref} sample, recovers $r^\star$'s ranking, which supports the premise of our attack.

L2: the ranked candidates. At the second exposure level, the attacker additionally observes all N candidates sampled by BoN and their ranking under $r^\star$, written $(y_{(1)}, \ldots, y_{(N)})$ from most to least preferred as in §3. Such information can be exposed by reranking APIs, which return candidates ordered according to a learned scoring model.[2] Unlike L1, where the attacker must construct comparison samples independently from π_{ref}, L2 directly reveals the relative preferences of $r^\star$ among the candidates actually considered by BoN. We exploit this richer supervision by fitting the observed ranking with a weighted form of the ListMLE loss (3) (Xia et al., 2008):

$$\ell_{\text{L2}} \;=\; -\sum_{k=1}^{N-1} w_k \log \frac{\exp \hat{r}_\phi(x, y_{(k)})}{\sum_{j=k}^{N} \exp \hat{r}_\phi(x, y_{(j)})}. \tag{8}$$

The weights let us study variants of this loss. Setting $w = (1, 0, \ldots, 0)$ keeps only the first term, so the shadow only learns that the winner $y^\star = y_{(1)}$ beats the other candidates. This case resembles the L1 loss, except that $r^\star$ itself certified $y_{(2)}, \ldots, y_{(N)}$ to lose to $y_{(1)}$, whereas L1 compares the winner against fresh π_{ref} samples. Setting $w = (1, \ldots, 1)$ recovers the original ListMLE loss. More weightings are studied in App. E.2.

L3: the scores. At the highest exposure level, the attacker additionally observes the scalar score (or logit) $r^\star(x, y_j)$ assigned to each BoN candidate, as can occur when a hosted reward-model endpoint directly returns scores.[3] Since these scores already induce the ranking available at L2, the additional information in L3 lies in their numerical values: score differences reveal not only which candidate $r^\star$ prefers, but also the strength of those preferences. We therefore ask whether access to this richer signal enables a more effective extraction attack. We adopt a cross-entropy loss on $\hat{r}_\phi$'s logits, so that the shadow directly copies $r^\star$'s predictions:

$$\ell_{\text{L3}} \;=\; -\frac{1}{P}\sum_{i<j}\Big[p^\star_{ij} \log \sigma\big(\hat{r}_\phi(x, y_i) - \hat{r}_\phi(x, y_j)\big) + (1 - p^\star_{ij}) \log \sigma\big(\hat{r}_\phi(x, y_j) - \hat{r}_\phi(x, y_i)\big)\Big], \tag{9}$$

where $p^\star_{ij} = \sigma\big(r^\star(x, y_i) - r^\star(x, y_j)\big)$ is $r^\star$'s preference and $P = \binom{N}{2}$ is the number of pairs.

4.2 Defense Attempt: Perturbing Winner Selection

The attacks above exploit the fact that BoN outputs carry information about the reward model used to select them. We take a first step toward mitigating this leakage by perturbing the selection process so that the served output is less coupled to the exact ordering induced by $r^\star$. We study two defenses.

[2] For example, Cohere Rerank returns input documents sorted by relevance: https://docs.cohere.com/reference/rerank.

[3] For example, NVIDIA serves Nemotron reward models through a scoring endpoint: https://build.nvidia.com/nvidia/llama-3_1-nemotron-70b-reward.

Soft-BoN. Instead of deterministically serving the highest-scoring candidate, Soft-BoN samples the served candidate from a softmax over $r^\star$'s scores:

$$\Pr[\tilde{y} = y_j] = \frac{\exp\left(\beta\, r^\star(x, y_j)\right)}{\sum_{l=1}^{N} \exp\left(\beta\, r^\star(x, y_l)\right)}. \tag{10}$$

As $\beta \to \infty$, this recovers standard BoN. Decreasing β increases the randomness of selection and moves the output distribution closer to π_{ref} (Verdun et al., 2025).

Gaussian score perturbation. Alternatively, we add Gaussian noise to each candidate's score before taking the argmax:

$$\tilde{r}_j = r^\star(x, y_j) + \xi_j, \qquad \xi_j \sim \mathcal{N}\big(0, (c\, s)^2\big), \qquad \tilde{y} = \arg\max_j \tilde{r}_j. \tag{11}$$

Here s is a fixed score scale and c controls the noise strength, with both specified in App. D. This strategy is related to output perturbation previously studied as a defense against model extraction (Orekondy et al., 2020; Kariyappa & Qureshi, 2020), as well as the Report Noisy Max mechanism in the differential privacy literature (Durfee & Rogers, 2019).

Both defenses trade some of BoN's selection quality for a noisier preference signal. In §5.4, we evaluate this privacy–utility tradeoff by measuring how much each defense weakens the attack relative to the corresponding reduction in BoN reward gain.

5 Experiments

5.1 Setup

Reward models, data, and shadow model. We use five open-weight reward models as $r^\star$. Skywork-Reward-Llama-3.1-8B (Skywork-RM) is trained on Skywork-Reward-Preference-80K-v0.1 dataset (Liu et al., 2024). Two further RMs are trained on the same dataset. Skywork-Reward-Gemma-2-27B (Gemma-27B-RM) (Liu et al., 2024) is built on Gemma-2-27B, and URM-LLaMa-3.1-8B (URM-8B) (Lou et al., 2024) is fine-tuned from Skywork-RM itself. Llama-3.1-Tulu-3-8B-RM (Tulu-RM) (Lambert et al., 2025) is trained on the 273K pairs of the Tulu 3 preference mixture. Llama-3.1-8B-Instruct-RM-RB2 (RB2-RM) (Malik et al., 2026) is trained on the Tulu mixture plus 77K Skywork v0.2 pairs. Skywork-RM and RB2-RM are fine-tuned from Llama-3.1-8B-Instruct, and Tulu-RM from Llama-3.1-Tulu-3-8B-SFT. We use Llama-3.1-8B-Instruct (Grattafiori et al., 2024) as the base policy π_{ref} for all five RMs. For each RM, we sample prompts from its own training dataset: 10,000 for the three Skywork-trained RMs, which share the same prompts and candidate responses, 27,290 for Tulu-RM, and 34,992 for RB2-RM, up to one-eighth of each RM's training data. This size keeps the attack within our compute budget. We also conduct the attack on domain-specific data outside $r^\star$'s training distribution. We take 10,000 math problems from UltraInteract (Yuan et al., 2025) and 10,000 coding problems from CodeContests (Li et al., 2022), then repeat the attack on Skywork-RM as above.

For every prompt, we draw $N = 8$ candidate responses from π_{ref} and let $r^\star$ score them, yielding the winner $y^\star$ for the L1 attacker, the ranked candidates for L2, and the scores for L3. At L1 the attacker also draws $K = N - 1 = 7$ fresh references per prompt from π_{ref}, so that the estimate of the L1 objective (6) mirrors a BoN draw with one winner against $N - 1$ other candidates. The shadow model $\hat{r}_\phi$ is Llama-3.1-8B-Instruct with LoRA adapters of rank 64 (Hu et al., 2022) and a linear score head, trained for three epochs on fewer than 1% of its parameters (App. C). A language model with a linear score head is one of the most common reward model architectures (Stiennon et al., 2020; Ouyang et al., 2022; Liu et al., 2024), and so is the architecture of the five RMs above.

Evaluation. The evaluation metric is ordering agreement, the fraction of pairs that the shadow model orders correctly:

$$\text{acc}(\hat{r}_\phi) \;=\; \frac{1}{|\mathcal{D}|} \sum_{(x, y_c, y_r) \in \mathcal{D}} \mathbf{1}\big[\hat{r}_\phi(x, y_c) > \hat{r}_\phi(x, y_r)\big], \tag{12}$$

where x is a prompt from $r^\star$'s training dataset that the attacker never used to train the shadow model, and y_c and y_r are two responses to x of which y_c is the preferred one. We evaluate on two sets of such

Table 1: Ordering agreement of the shadow model with $r^\star$ at each exposure level (mean$_{\pm \text{s.d.}}$ over five seeds). Each RM has its own evaluation sets, so values are compared within a row; near-tie pairs ($|\Delta r^\star| < 0.25\sigma$) are excluded from every RM's curated set. For the math and code prompts the curated pairs are the datasets' own pairs, which $r^\star$ was not trained on.

		curated pairs			generated pairs		
RM	random	L1	L2	L3	L1	L2	L3
Skywork-RM	0.5	$0.754_{\pm 0.051}$	$0.942_{\pm 0.003}$	$0.942_{\pm 0.006}$	$0.829_{\pm 0.058}$	$0.979_{\pm 0.004}$	$0.983_{\pm 0.002}$
Tulu-RM	0.5	$0.704_{\pm 0.017}$	$0.967_{\pm 0.002}$	$0.974_{\pm 0.003}$	$0.750_{\pm 0.032}$	$0.963_{\pm 0.003}$	$0.970_{\pm 0.003}$
RB2-RM	0.5	$0.631_{\pm 0.025}$	$0.953_{\pm 0.002}$	$0.959_{\pm 0.001}$	$0.741_{\pm 0.040}$	$0.974_{\pm 0.001}$	$0.980_{\pm 0.001}$
URM-8B	0.5	$0.746_{\pm 0.059}$	$0.945_{\pm 0.003}$	$0.952_{\pm 0.005}$	$0.839_{\pm 0.056}$	$0.981_{\pm 0.002}$	$0.984_{\pm 0.001}$
Gemma-27B-RM	0.5	$0.738_{\pm 0.049}$	$0.937_{\pm 0.003}$	$0.939_{\pm 0.003}$	$0.850_{\pm 0.055}$	$0.985_{\pm 0.001}$	$0.985_{\pm 0.002}$
Skywork-RM							
math prompts	0.5	$0.746_{\pm 0.073}$	$0.980_{\pm 0.001}$	$0.984_{\pm 0.002}$	$0.884_{\pm 0.030}$	$0.989_{\pm 0.002}$	$0.993_{\pm 0.001}$
code prompts	0.5	$0.664_{\pm 0.085}$	$0.909_{\pm 0.006}$	$0.903_{\pm 0.003}$	$0.797_{\pm 0.053}$	$0.966_{\pm 0.001}$	$0.960_{\pm 0.004}$

pairs, curated and generated, which differ only in the source of the response pair (y_c, y_r). A *curated* pair is the (chosen, rejected) response pair for x recorded in $r^\star$'s training dataset, so agreement on curated pairs tests whether the shadow model has learned the preference behind $r^\star$'s training dataset, which might pose a threat to the users and annotators. A *generated* pair is two responses sampled from π_{ref}, and y_c is the one $r^\star$ scores higher, so agreement on generated pairs tests whether the shadow model behaves like $r^\star$ on unseen responses, which might threaten the RM's owner. Both sets drop near-tie pairs, whose two $r^\star$ scores differ by less than a quarter of the standard deviation of $r^\star$'s scores, since $r^\star$ itself barely separates them. We exclude them because they carry less privacy risk, and we focus on the more vulnerable pairs; the unfiltered results and the choice of the threshold are reported in App. C as well.

Every number in the tables is a mean over five training seeds with its standard deviation, unless stated otherwise. Two further measurements appear later. §5.3 scores the shadow models on RewardBench-2, and §5.4 measures the tradeoff between defense protection and RM's utility.

5.2 How much each exposure level leaks

Table 1 shows that the L1 attacker already recovers 63% to 75% of the preference behind each RM's training data, and on the generated set it further agrees with $r^\star$ on 74% to 85% of the pairs. The shadow model does better on generated pairs because these responses are sampled from π_{ref}, and the shadow model itself is also trained on fresh samples from π_{ref}. Nevertheless, the shadow model still recovers much of the preference on the curated set, up to 75%, well above the 50% of chance. L2 goes a step further and raises the agreement to 94% to 97% on curated pairs and 96% to 99% on generated pairs, and the attack becomes more stable with a standard deviation across seeds below 0.01. This confirms that the non-winning candidates and their ranking bring a gain of 0.14 to 0.32 in agreement to the shadow model. We also study how much the attacker loses when it knows only part of the ranking, as an ablation in App. E.2. L3, however, moves the agreement by at most 0.007 compared to L2. The ranking therefore brings the attacker most of the gain, and knowing how far apart $r^\star$ places the candidates adds little on top of it. On the math and code prompts the pattern holds. L1 recovers 88% and 80% of the generated pairs, and the L2 raises the agreement to 99% and 97%. Gemma-27B-RM's backbone differs from the shadow model's in family and size, yet still leaks as much as Skywork-RM. BoN still leaks $r^\star$'s preferences even when the attacker's prompts lie outside $r^\star$'s training data or its shadow model does not share $r^\star$'s backbone. More ablations on alternative losses are in App. B and per-seed values in App. E.4.

5.3 The shadow model on RewardBench-2

RewardBench-2 (Malik et al., 2026) evaluates a reward model's ability to tell a correct response from flawed ones, with $1{,}865$ prompts across six skill categories. A rejected response carries one flaw, such as subtle factual error or off-topic reply. Each prompt has one chosen and three rejected responses, except in the Ties category, which has 19 responses on average, so a random scorer gets

Table 2: Accuracy on RewardBench-2 and agreement with $r^\star$ (mean over five seeds, s.d. as subscript). Agreement with $r^\star$ is the ordering agreement (12) on RewardBench-2's chosen-versus-rejected pairs with $r^\star$'s scores as the labels, so it also counts the pairs that $r^\star$ gets wrong. A random scorer gets 0.24 accuracy and 0.5 agreement.

	RewardBench-2 accuracy				agreement with $r^\star$		
RM	L1	L2	L3	$r^\star$	L1	L2	L3
Skywork-RM	$0.284_{\pm 0.119}$	$0.598_{\pm 0.016}$	$0.620_{\pm 0.025}$	0.805	$0.593_{\pm 0.124}$	$0.900_{\pm 0.006}$	$0.907_{\pm 0.004}$
Tulu-RM	$0.249_{\pm 0.003}$	$0.543_{\pm 0.010}$	$0.557_{\pm 0.009}$	0.631	$0.563_{\pm 0.038}$	$0.911_{\pm 0.006}$	$0.926_{\pm 0.003}$
RB2-RM	$0.239_{\pm 0.006}$	$0.624_{\pm 0.020}$	$0.633_{\pm 0.047}$	0.792	$0.497_{\pm 0.046}$	$0.918_{\pm 0.005}$	$0.928_{\pm 0.018}$

about 0.24 of total accuracy. We use this benchmark to test the utility of the shadow model to the attacker. It asks how well the shadow model can replace $r^\star$ as a reward model on general tasks beyond $r^\star$'s training data, which might threaten the RM's owner. Table 2 reports two measures. RewardBench-2 accuracy measures the shadow model's utility, and agreement with $r^\star$ measures how closely the shadow model copies $r^\star$'s behavior, even on the pairs that $r^\star$ gets wrong. Per-category accuracies and two likelihood baselines from π_{ref} are in App. E.5.

The L2 and L3 shadow models reach 0.54 to 0.63 accuracy, and agree with $r^\star$ on 90% to 93% of the pairs. This benchmark is hard even for $r^\star$, which reaches only 0.63 to 0.81 RewardBench-2 accuracy itself. The L2 and L3 attackers therefore hold an approximate copy of $r^\star$. It agrees with $r^\star$ on 96% to 98% of the generated pairs (Table 1) and generalizes to the prompts and responses of RewardBench-2. The L1 attack does not reach that far. L1 shadow models stay near chance on both measures, at 0.24 to 0.28 accuracy and 50% to 59% agreement. Yet on the curated pairs of Table 1, the L1 shadow models still recover 63% to 75% of those preferences. The threat at L1 therefore comes mainly from recovering the preferences behind $r^\star$'s training data, while RM distillation to replace the true RM $r^\star$ is also a threat under L2 and L3.

5.4 Perturbing the winner selection

As a first step toward a defense, we evaluate the two defenses of §4 on Skywork-RM (Figure 2). A defense costs the deployment part of BoN's utility. We measure this cost as retained gain,

$$\text{retained gain} \;=\; \frac{\mathbb{E}[r^\star(x,\tilde{y})] - \mathbb{E}[r^\star(x,y)]}{\mathbb{E}[r^\star(x,y^\star)] - \mathbb{E}[r^\star(x,y)]}, \tag{13}$$

where $\tilde{y}$ is the response the defended deployment serves, $y^\star$ is the true BoN winner, and y is a uniformly random candidate. Retained gain is 1 for undefended BoN and 0 for random serving. Gaussian noise has strength c, the noise level relative to the spread of the scores, and soft-BoN has strength β, the temperature of its softmax. A stronger defense (larger c or smaller β) moves the served response further from the true winner and thus costs more utility. The two strengths are not comparable, so we compare the defenses at equal retained gain, with each β chosen to match one c (App. D). The attacker does not know the defense and runs the attack of §4 unchanged on what the defended deployment serves. Each panel of Figure 2 plots the ordering agreement on the generated pairs against retained gain for one exposure level, five seeds per point. We do not evaluate the L3 attack against soft-BoN, because the attacker still receives the true score of every candidate. Results on curated pairs and more details are in App. E.6.

Both defenses weaken the attack only slightly. When the deployment gives up 27% of BoN's gain, agreement falls from 0.83 to 0.76 at L1, from 0.98 to 0.93 at L2, and from 0.98 to 0.94 at L3 under Gaussian noise, and by less under soft-BoN. When it gives up half of the gain, so that the served response is not the true winner on about 70% of the prompts, agreement still stays at 0.74 at L1, 0.84 at L2, and 0.89 at L3. None of these points comes near the 0.5 of chance. Perturbing the winner selection therefore trades utility for a slightly weaker attack and does not close the leak.

6 Final Remarks

BoN keeps a private reward model $r^\star$ in the serving loop, and every served response reveals a choice $r^\star$ made. We showed that this side channel is exploitable. A shadow model trained on served winners

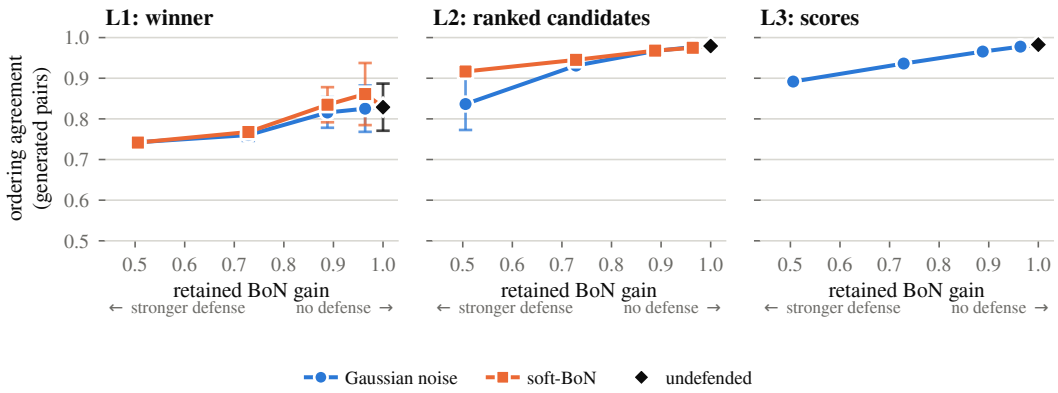

Figure 2: Ordering agreement of the Skywork-RM shadow model on generated pairs against retained gain (undefended = 1). Different points correspond to different defense parameters. Both defenses weaken the attack slightly unless with severe loss on retained gain.

alone (L1) recovers the preferences behind $r^\star$'s training data well above chance, which harms the users and annotators whose preferences the RM encodes. Once the deployment also returns the ranked candidates (L2), the shadow model becomes an approximate copy of $r^\star$ and agrees with it even on prompts far from its own training data, which harms the RM's owner as well. Returning the scores (L3) adds little beyond the ranking. Perturbing the winner selection weakens the attack only slightly and costs part of BoN's gain.

Future directions.

- **A better defense at inference time.** An immediate next step is to develop a defense against BoN's leak of preferences at inference time. Our experiments show that output perturbation does not work well, so a defense might have to cut the link between the served response and $r^\star$'s choice while keeping most of BoN's gain.
- **Training the RM against the leak.** The RM itself can also be trained to avoid this leak. Differentially private reward learning limits what $r^\star$ reveals about any single comparison, and whether this training method stops a shadow model from copying $r^\star$'s ordering remains underexplored.
- **A stronger attacker for distilling $r^\star$.** Our attack is a first attempt to exploit BoN and fits one objective per exposure level. Whether a better objective or a different technique steals more of $r^\star$'s preferences should be tested next.
- **A stronger membership inference of $r^\star$ through shadow model.** Our shadow model also raises a membership question. It recovers the pairs behind $r^\star$'s training data, so whether the served winners also reveal which comparisons were in that data is worth testing.
- **What the L1 shadow model learns.** The L1 shadow model recovers the preferences behind $r^\star$'s training data yet stays near chance on RewardBench-2 (§5.3). Which part of $r^\star$'s preference it captures, and why that part does not carry over to new prompts, is open.
- **Privacy vulnerability of other inference-time alignment mechanisms.** Reward-guided decoding (Khanov et al., 2024; Mudgal et al., 2024) and search with a process reward model (Snell et al., 2025) also keep a reward model in the serving loop. Whether our attack transfers to them, and how much each one leaks, should be measured next.
- **Theoretical privacy-utility trade-off analysis.** BoN's winner distribution has a closed form (§3), so the leak can be analyzed rather than only measured. How much a served response reveals about $r^\star$ as a function of N and the defense strength, and the best trade-off between that leak and retained gain that any selection rule can reach, are open questions.

AI USE STATEMENT

In this work we used generative AI tools, namely the Claude Code and Codex coding assistants, for the following tasks with required disclosure. Generating synthetic datasets: every attack corpus and every generated evaluation pair consists of responses sampled from the base policies Llama-3.1-8B-Instruct and Llama-3.1-8B, which are the systems under study, and the code that builds, filters, and freezes these corpora was written by the assistants from the authors' specifications. Implementing methods: most of the experiment code, including the data pipeline, the three shadow-model trainers, the evaluation and consolidation scripts, the defense builders, and the smoke test, was written by the assistants and then read, run, and debugged by the authors. Research ideation and execution: the authors formulated the research question, the threat model, the three exposure levels, the losses, and the defenses; the assistants turned the authors' list of experiments into an execution plan, proposed some controls, such as the backbone control in App. E.3 that followed from an internal review, launched and monitored the training runs, and flagged places where a result did not support a claim. Mathematical claims: the authors derived the L1 objective from the BoN winner distribution (§4.1 and App. A), and the assistants helped organize the derivation and checked its steps. Interpreting results: the assistants computed the summary statistics from the run artifacts and proposed candidate explanations for some results, which the authors accepted, rejected, or rewrote; every interpretation in the paper was decided by the authors. We also discussed the conceptual framework of the paper with the assistants while developing it, and the framing decisions were the authors' own. Qualitative or thematic data analysis is not applicable to this work.

Additionally, we used generative AI tools for tasks with recommended disclosure: drafting sections of the paper from the authors' outlines and comments; suggesting the structure of sections; searching for and summarizing related literature; formatting and checking references; writing the plotting scripts for the figures and drafts of the teaser (Figure 1); and producing internal reviews of the draft in the style of conference reviewers.

We have reviewed all AI-assisted work. All assistant-written code was read and tested by the authors, and every number in the paper is recomputed from the released run artifacts by the released scripts. Every citation was checked by the authors against the cited work, and the claims the paper makes about prior work were verified against the full text of each cited paper. The derivation in §4.1 was verified step by step by the authors. All assistant-drafted text was revised by the authors, and its claims were checked against the experiments. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work exposes a privacy and intellectual-property risk in inference-time alignment. Using Best-of-N as the representative case, we show that the responses a BoN deployment serves carry preference signals from its private reward model, and that an attacker who only observes those responses can recover the preferences behind the reward model's training data and distill an approximate copy of the model. Our purpose is to unveil this risk rather than to exploit it: deployers of BoN and related methods are better served by knowing that the winner alone acts as a side channel than by leaving it unexamined. All experiments use open-weight reward models (Skywork-RM, Tulu-RM, RB2-RM, URM-8B, and Gemma-27B-RM), an open base policy, and publicly released preference datasets. No proprietary weights, private reward scores, or personally identifying information were accessed or released, no new human preference data were collected, and the attack never queries the target reward model. We report the leakage as a lower bound from a deliberately simple attack so as not to overstate what an adaptive adversary could achieve. We also take a first step toward a defense by perturbing the winner selection. Our results show that the risk remains severe: the attack survives almost intact unless the deployment gives up a substantial share of BoN's reward gain. We therefore do not recommend any of the studied mechanisms as sufficient protection on their own, and we hope this work motivates further study of defenses that close the leak without sacrificing utility.

REPRODUCIBILITY STATEMENT

All code, frozen inputs and run artifacts are released in a single repository (https://anonymous.4open.science/r/Preference-Stealing-BoN-B20C/; the de-anonymized repository is made public on acceptance): the shadow-model package and the three trainers, the data pipeline, one entry script per experiment family, and every run's per-epoch evaluation log, per-pair test and curated dumps and completion markers, so that every table and figure is recomputable on a CPU from the committed artifacts (`consolidate_ladder.py`, `aggregate_curated.py`, `rewardbench2_correlate.py`, `fig_defense.py`); `experiments/EXPERIMENTS.md` lists every run behind the paper in execution order with its command, cost and output, and the README maps each table and figure to the artifact it was computed from. The frozen corpora are hash-pinned: the Skywork corpus (training prompts with their $N{=}8$ candidates and scores, fresh π_{ref} references, evaluation and validation pairs, and the curated set) ships as one tarball whose SHA-256 and per-file hashes are re-verified by the download script; the defended and N-sweep corpora are rebuilt deterministically from it and the builder aborts unless the rebuilt tournament reproduces the pinned hash; the MATH and CODE corpora and every curated set ship the same way; the Tulu-RM and RB2-RM corpora are regenerated from the frozen prompt splits with the released generation scripts. Model weights are fetched from the Hugging Face Hub and used offline. The shadow model's training recipe and its differences from Skywork-RM's are given in App. C (Table 5); the per-level losses are derived in §4 and App. A, with alternatives in App. B. Data construction (prompts, BoN sampling settings, near-tie filtering, and the evaluation sets) is described in §5.1 and App. C, the metric (ordering agreement) is defined in §5.1, the defense parameters are in App. D and every run's value in App. E.4. Tables report the mean and sample standard deviation over five training seeds unless a cell is marked as one seed, with the epoch selected on a validation split of generated pairs. A smoke test (`experiments/code/smoke.sh`) drives every entry script on tiny inputs and asserts every artifact; its CPU part passes from a fresh clone after the download scripts.

REFERENCES

Y. Bai, S. Kadavath, S. Kundu, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

B. Brown et al. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

N. Carlini et al. Stealing part of a production language model. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2403.06634.

Chen Chen, Yaolin Chen, Xuehan Sun, Juan Lin, Xueluan Gong, Yuhang Zheng, Qian Wang, and Kwok-Yan Lam. JudgeStealer: Extracting LLM judging capabilities across evaluation protocols. *arXiv preprint arXiv:2608.26982*, 2026.

Y. Chen et al. Hard-label cryptanalytic extraction of neural network models. In *ASIACRYPT*, 2024. arXiv:2409.11646.

K. Cobbe et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

David Durfee and Ryan M. Rogers. Practical differentially private top-k selection with pay-what-you-get composition. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. arXiv:1905.04273.

Qizhang Feng, Siva Rajesh Kasa, Santhosh Kumar Kasa, Hyokun Yun, Choon Hui Teo, and Sravan Babu Bodapati. Exposing privacy gaps: Membership inference attack on preference data for LLM alignment. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 258 of *PMLR*, pp. 5221–5229, 2025. arXiv:2407.06443.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning (ICML)*, volume 202 of *PMLR*, pp. 10835–10866, 2023. arXiv:2210.10760.

A. Grattafiori, A. Dubey, A. Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2106.09685.

M. Jagielski et al. High accuracy and high fidelity extraction of neural networks. In *USENIX Security Symposium*, 2020. arXiv:1909.01838.

Jared Joselowitz, Ritam Majumdar, Arjun Jagota, Matthieu Bou, Nyal Patel, Satyapriya Krishna, and Sonali Parbhoo. Insights from the inverse: Reconstructing LLM training goals through inverse reinforcement learning. In *Conference on Language Modeling (COLM)*, 2025. arXiv:2410.12491.

Sanjay Kariyappa and Moinuddin K. Qureshi. Defending against model stealing attacks with adaptive misinformation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. arXiv:1911.07100.

Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. ARGS: Alignment as reward-guided search. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2402.01694.

K. Krishna et al. Thieves on sesame street! model extraction of BERT-based APIs. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:1910.12366.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Conference on Language Modeling (COLM)*, 2025. arXiv:2411.15124.

H. Lee, S. Phatale, H. Mansoor, T. Mesnard, J. Ferret, K. Lu, C. Bishop, E. Hall, V. Carbune, A. Rastogi, and S. Prakash. RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2309.00267.

Y. Li, D. Choi, J. Chung, et al. Competition-level code generation with AlphaCode. *Science*, 378 (6624):1092–1097, 2022. arXiv:2203.07814.

C. Y. Liu et al. Skywork-Reward: Bag of tricks for reward modeling in LLMs. *arXiv preprint arXiv:2410.18451*, 2024.

J. Liu, Y. Kang, D. Tang, K. Song, C. Sun, X. Wang, W. Lu, and X. Liu. Order-disorder: Imitation adversarial attacks for black-box neural ranking models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022. arXiv:2209.06506.

T. Liu et al. LiPO: Listwise preference optimization through learning-to-rank. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2025a. arXiv:2402.01878.

Yule Liu, Heyi Zhang, Jinyi Zheng, Zhen Sun, Zifan Peng, Jiaheng Wei, Tianshuo Cong, Yilong Yang, and Xinlei He. Auditing data membership in reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2511.14045*, 2025b. v2, May 2026.

Xingzhou Lou, Dong Yan, Wei Shen, Yuzi Yan, Jian Xie, and Junge Zhang. Uncertainty-aware reward model: Teaching reward models to know what is unknown. *arXiv preprint arXiv:2410.00847*, 2024.

R. D. Luce. *Individual Choice Behavior: A Theoretical Analysis*. Wiley, New York, 1959.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. RewardBench 2: Advancing reward model evaluation. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2506.01937.

N. Mireshghallah, M. Antoniak, Y. More, Y. Choi, and G. Farnadi. Trust no bot: Discovering personal disclosures in human-LLM conversations in the wild. In *Conference on Language Modeling (COLM)*, 2024. arXiv:2407.11438.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, Jilin Chen, Alex Beutel, and Ahmad Beirami. Controlled decoding from language models. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2310.17022.

R. Nakano et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2000.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against DNN model stealing attacks. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:1906.10908.

L. Ouyang et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2203.02155.

N. Patel, M. Bou, A. Jagota, S. Krishna, and S. Parbhoo. Learning from failures: Understanding LLM alignment through failure-aware inverse RL. *arXiv preprint arXiv:2510.06092*, 2025.

R. L. Plackett. The analysis of permutations. *Applied Statistics*, 24(2):193–202, 1975.

R. Pukdee, M.-F. Balcan, and P. Ravikumar. Reward learning from best-of-n preference data: Targets, tradeoffs, and design principles. *arXiv preprint arXiv:2605.30619*, 2026.

Sayak Ray Chowdhury, Xingyu Zhou, and Nagarajan Natarajan. Differentially private reward estimation with preference feedback. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 238 of *PMLR*, pp. 4843–4851, 2024. arXiv:2310.19733.

S. Sanyal, S. Addepalli, and R. V. Babu. Towards data-free model stealing in a hard label setting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. arXiv:2204.11022.

P. G. Sessa et al. BOND: Aligning LLMs with best-of-n distillation. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2407.14622.

R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy (S&P)*, 2017. arXiv:1610.05820.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2408.03314.

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 (NeurIPS)*, volume 33, 2020. arXiv:2009.01325.

H. Touvron et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Stealing machine learning models via prediction APIs. In *USENIX Security Symposium*, 2016.

Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio P. Calmon. Soft best-of-n sampling for model alignment. In *IEEE International Symposium on Information Theory (ISIT)*, pp. 1–6, 2025. doi: 10.1109/ISIT63088.2025.11195541. arXiv:2505.03156.

S. Welleck et al. From decoding to meta-generation: Inference-time algorithms for large language models. *Transactions on Machine Learning Research (TMLR)*, 2024. arXiv:2406.16838.

F. Wu, H. A. Inan, et al. Privately aligning language models with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.16960.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: Theory and algorithm. In *International Conference on Machine Learning (ICML)*, pp. 1192–1199, 2008. doi: 10.1145/1390156.1390306.

Joy Qiping Yang, Salman Salamatian, Ziteng Sun, Ananda Theertha Suresh, and Ahmad Beirami. Asymptotics of language model alignment. In *IEEE International Symposium on Information Theory (ISIT)*, pp. 2027–2032, 2024. doi: 10.1109/ISIT57864.2024.10619456. arXiv:2404.01730.

H. Yao et al. Black-box guardrail reverse-engineering attack. *arXiv preprint arXiv:2511.04215*, 2025.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Boji Shan, Zeyuan Liu, Jia Deng, Huimin Chen, Ruobing Xie, Yankai Lin, Zhenghao Liu, Bowen Zhou, Hao Peng, Zhiyuan Liu, and Maosong Sun. Advancing LLM reasoning generalists with preference trees. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2404.02078.

Zhenrui Yue, Zhankui He, Huimin Zeng, and Julian McAuley. Black-box attacks on sequential recommenders via data-free model extraction. In *ACM Conference on Recommender Systems (RecSys)*, 2021. arXiv:2109.01165.

Q. Zhang, F. Lyu, Z. Sun, et al. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025.

K. Zhao, L. Li, K. Ding, N. Z. Gong, Y. Zhao, and Y. Dong. A survey on model extraction attacks and defenses for large language models. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2025. arXiv:2506.22521.

L. Zheng et al. LMSYS-Chat-1M: A large-scale real-world LLM conversation dataset. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.11998.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023. arXiv:2306.05685.

Banghua Zhu, Michael I. Jordan, and Jiantao Jiao. Principled reinforcement learning with human feedback from pairwise or K-wise comparisons. In *International Conference on Machine Learning (ICML)*, volume 202 of *PMLR*, pp. 43037–43067, 2023. arXiv:2301.11270.

B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2008.

A BoN WINNER LIKELIHOOD

With $y_1, \ldots, y_N \sim \pi_{\text{ref}}(\cdot \mid x)$ and $y^\star = \arg\max_i r(x, y_i)$, for a fixed y, $\Pr[y^\star = y] = N\,\pi_{\text{ref}}(y)\,F_r(x,y)^{N-1}$ where $F_r(x,y) = \Pr_{y' \sim \pi_{\text{ref}}}[r(x,y') < r(x,y)]$. Logging and dropping ϕ-independent terms gives $\max_\phi \sum_i \log F_\phi(x_i, y_i^\star)$. With N fixed, $(N{-}1)$ is constant in ϕ; N enters through the data because the winner is the maximum of N draws, with $\mathbb{E}[F_r(x,y^\star)] = N/(N{+}1)$.

B ALTERNATIVE LOSSES

Table 3 reports two losses that the main text does not use. At L1, ℓ_{BT} is the Bradley–Terry loss (2) with one fresh π_{ref} sample as the rejected response, which is ℓ_{L1} at $K = 1$. At L3, ℓ_{MSE} regresses the shadow model's score on $r^\star$'s score of every candidate,

$$\ell_{\text{MSE}} = \frac{1}{N} \sum_{j=1}^{N} \big(\hat{r}_\phi(x, y_j) - r^\star(x, y_j)\big)^2. \tag{14}$$

The two L1 losses are indistinguishable over seeds on every RM, and ℓ_{L1} varies more across seeds. ℓ_{MSE} trails ℓ_{L2} and ℓ_{L3} on every RM.

Table 3: Ordering agreement under the alternative losses, next to the losses of the main text (mean over five seeds, s.d. as subscript). ℓ_{BT} is ℓ_{L1} with one fresh reference; ℓ_{MSE} regresses on $r^\star$'s scores.

		Skywork-RM		Tulu-RM	RB2-RM
level	loss	curated	generated	generated	generated
L1	ℓ_{BT} (K=1)	$0.711_{\pm 0.039}$	$0.787_{\pm 0.048}$	$0.755_{\pm 0.004}$	$0.758_{\pm 0.003}$
L1	ℓ_{L1} (K=7)	$0.754_{\pm 0.051}$	$0.829_{\pm 0.058}$	$0.745_{\pm 0.030}$	$0.740_{\pm 0.040}$
L2	ℓ_{L2}	$0.942_{\pm 0.003}$	$0.979_{\pm 0.004}$	$0.964_{\pm 0.003}$	$0.972_{\pm 0.004}$
L3	ℓ_{L3}	$0.942_{\pm 0.006}$	$0.983_{\pm 0.002}$	$0.969_{\pm 0.002}$	$0.976_{\pm 0.006}$
L3	ℓ_{MSE}	$0.924_{\pm 0.013}$	$0.971_{\pm 0.009}$	$0.961_{\pm 0.002}$	$0.958_{\pm 0.016}$

C SETUP DETAILS

Attack data. Prompts come from each RM's training data. For Skywork-RM they are 10,000 unique single-turn prompts of Skywork-Reward-Preference-80K-v0.1, with the HelpSteer2 rows excluded. For Tulu-RM they are one tenth of the Tulu 3 preference mixture, stratified by source. For RB2-RM they are the same 27,290 Tulu prompts, rescored by RB2-RM, plus 7,702 prompts from Skywork-Reward-Preference-80K-v0.2. URM-8B and Gemma-27B-RM reuse Skywork-RM's prompts, candidates, and fresh references, rescored by each RM. The math prompts are 10,000 problems from MATH and GSM8K, with a correct and an incorrect solution from UltraInteract as the curated pair; the code prompts are 10,000 CodeContests problems, with an accepted and a rejected human submission as the curated pair. For every prompt π_{ref} generates $N = 8$ candidates through its chat template (temperature 1, top-p 1, at most 1,024 new tokens) and $r^\star$ scores them. Truncated responses are kept; 1.9% of the Skywork-RM winners are truncated. The attacker's fresh references are eight further π_{ref} samples per prompt, drawn once and frozen; the L1 attack uses seven of them. The data are fixed across seeds.

Evaluation sets. Each RM has 3,000 held-out prompts and a further 500 for validation. A generated pair is two candidates of the same prompt, labeled by $r^\star$, and a curated pair is the dataset's own (chosen, rejected) pair for that prompt, relabeled by $r^\star$; for Skywork-RM, $r^\star$ agrees with the dataset's own labels on 99.9% of these pairs. Both sets drop pairs whose $r^\star$ scores differ by less than $\tau = 0.25\,\sigma$, where σ is the standard deviation of $r^\star$'s scores over the generated candidates. This leaves 1,987, 1,743, 1,922, 1,557, and 1,664 generated test pairs for Skywork-RM, Tulu-RM, RB2-RM, URM-8B, and Gemma-27B-RM, 2,221 and 1,618 for the math and code prompts, and validation sets of 339, 304, 318, 270, 277, 358, and 247 pairs. The curated sets keep 2,997 of 3,000 pairs for Skywork-RM, 1,276 of 1,771 for Tulu-RM, 1,647 of 1,999 for RB2-RM, 2,503 and 2,791 of 2,819 for URM-8B and Gemma-27B-RM, 2,600 of 2,833 for the math prompts, and 1,281 of 1,702 for the code prompts. Each RM has its own evaluation sets and its own τ, so accuracies are compared within an RM only.

Near-tie pairs. A pair whose two $r^\star$ scores differ by less than τ is a near-tie. $r^\star$ barely separates the two responses, so its choice between them reveals little of its preference and carries less privacy risk. We therefore report agreement on the pairs above τ, and we set τ to a quarter of the standard deviation of $r^\star$'s scores over the generated candidates so that the rule scales with each RM's score range. Agreement rises with the margin. On Tulu-RM, for example, L2 orders 75% of the curated pairs below τ correctly and 98% of the pairs above 5τ. The rule removes 34%, 42%, 36%, 26%, and 18% of the generated pairs of Skywork-RM, Tulu-RM, RB2-RM, and the math and code prompts, and between 1% and 28% of the curated pairs. Table 4 reports the curated agreement without the filter. It is lower by at most 0.07, and the ordering of the levels is unchanged. The generated sets were built with the filter, so no unfiltered generated agreement exists. The curated set of Skywork-RM was frozen with $\tau = 2.02$, computed from an earlier generation with a narrower score spread, which removes 3 of 3,000 pairs. At the $\tau = 6.23$ of the current generation 2,958 pairs remain, and agreement changes by at most 0.003.

Table 4: Curated agreement without the near-tie filter (mean$_{\pm \text{s.d.}}$ over five seeds), next to the share of pairs that the filter keeps. Table 1 reports the filtered values.

RM	pairs kept by the filter	L1	L2	L3
Tulu-RM	72%	$0.662_{\pm 0.015}$	$0.906_{\pm 0.003}$	$0.920_{\pm 0.007}$
RB2-RM	82%	$0.614_{\pm 0.023}$	$0.919_{\pm 0.003}$	$0.923_{\pm 0.002}$
URM-8B	89%	$0.734_{\pm 0.058}$	$0.931_{\pm 0.002}$	$0.940_{\pm 0.004}$
Gemma-27B-RM	99%	$0.737_{\pm 0.049}$	$0.934_{\pm 0.003}$	$0.936_{\pm 0.003}$
Skywork-RM, math prompts	92%	$0.728_{\pm 0.068}$	$0.956_{\pm 0.002}$	$0.961_{\pm 0.002}$
Skywork-RM, code prompts	75%	$0.625_{\pm 0.069}$	$0.834_{\pm 0.006}$	$0.832_{\pm 0.002}$

Shadow model. Table 5 lists the recipe next to Skywork-RM's. The backbone is frozen in bf16. LoRA adapters ($r = 64$, $\alpha = 128$) on the query, key, value, and output projections and a linear head in fp32 on the last token are trained with AdamW, a linear schedule with 5% warmup, gradient clipping at 1, and 16 prompts per step, for three epochs; the trainable parameters are 54.5M (0.73%). The input is the raw string `prompt+"\n"+response` with at most 1,536 tokens, and no chat template. The epoch with the best validation accuracy is evaluated. One run takes about 3 hours for ℓ_{BT} and 8 to 11 hours for the other losses on one 96 GB GPU.

Table 5: Training recipe of the shadow model next to Skywork-RM's (Skywork-RM numbers from Liu et al., 2024, §4.1.1). Above the rule: attacker budget. Below: threat-model constraints.

	Skywork-RM	shadow model
backbone	Llama-3.1-8B-Instruct	same (public)
tuning	full fine-tune	LoRA r=64, α=128 on q, k, v, o; base frozen
head	sequence-classification head, Linear$(4096 \to 1)$, no bias	same architecture, Linear$(4096 \to 1)$ in fp32, with bias; both read the last non-pad token
learning rate	2×10^{-6}	2×10^{-4} (LoRA) / 1×10^{-4} (head)
schedule	cosine	linear, 5% warmup
weight decay	10^{-3}	0
epochs	2	3
batch	global 128	16 prompts/step (grad. accum.)
loss	Bradley–Terry	ℓ_{L1}, ℓ_{L2}, ℓ_{L3} (§4)
input format	chat template (user/assistant)	raw `prompt+"\n"+response`
training data	80K preference pairs	BoN outputs on 10,000 prompts

D DEFENSE DETAILS

Both defenses of §4 act on $r^\star$'s scores for the N candidates of a prompt before the winner is chosen. Noise is drawn independently for every prompt and candidate.

The scale s. For each prompt we take the sample standard deviation of $r^\star$'s scores over its N candidates, which measures how far apart the candidates' scores typically are. The scale s is the mean of this quantity over the 10,000 prompts of the Skywork-RM corpus, $s = 12.59$, computed once on the undefended scores and then held fixed. The Gaussian noise in (11) has standard deviation $c\,s$, so c is dimensionless and c=1 adds noise as large as that typical spread.

Retained gain. A defense changes only which of the N candidates is served, and $r^\star$ has already scored every candidate, so the reward a defended deployment gives up can be computed on the frozen attack corpus without new generations or new scoring. Over its 10,000 prompts, a random candidate has mean $r^\star$ score 6.98 and the BoN winner 24.74. The retained gain of a deployment is the expected $r^\star$ score of the response it serves, minus 6.98, divided by the difference 17.76, so retained gain is 1 for BoN and 0 for random serving. The expectations in (13) run over the prompts of the attack corpus and over the randomness of the defense. The judge is $r^\star$ itself, which is the objective the deployment maximizes with BoN in the first place. For soft-BoN the expectation is exact, $\sum_j \Pr[\tilde{y} = y_j]\, r^\star(x, y_j)$ for each prompt. For Gaussian and Laplace noise it is a Monte Carlo

average over 400 draws of the noise per prompt, with a standard error below 0.0003. For random N it is exact through the order statistics of the eight scores. Gaussian noise at $c = 0.25$, 0.5, 1, and 2 retains 0.964, 0.888, 0.728, and 0.506.

Matching β to c. The strengths c and β are not comparable, so we compare the two defenses at equal retained gain. We keep the four values of c and choose a β for each one. Retained gain is monotone in β, so we evaluate it on a geometric grid of 240 values of β from 10^{-4} to 5, which needs no training, and interpolate to the retained gain of each c. This gives $\beta = 0.449$, 0.232, 0.121, and 0.063, whose retained gains match the Gaussian ones to three decimals (Table 6). Laplace noise with scale $b = 4.42$ and 8.84 matches $c = 0.5$ and 1 in the same way. Serving the winner of a uniformly random subset of $N \in \{2, 4, 8\}$ candidates retains 0.715.

Realized corpora. Each defended corpus that the attacker trains on realizes its rule once, with one draw of the noise or of the softmax per prompt under a fixed seed. Table 6 lists, next to the expected retained gain, the value realized in each corpus and the share of prompts whose served response is not the true winner. The realized values are within 0.012 of the expected ones, so the x axis of Figure 2 describes the corpora the shadow models were trained on.

Table 6: Retained gain of every defended deployment, expected under the defense rule and realized in the corpus the attacker trains on, with the share of prompts whose served response is not the true BoN winner.

		retained gain		winner replaced	
defense	strength	expected	realized	expected	realized
Gaussian noise	c=0.25	0.964	0.964	23.5%	23.4%
Gaussian noise	c=0.5	0.888	0.890	39.1%	38.7%
Gaussian noise	c=1	0.728	0.726	56.8%	57.5%
Gaussian noise	c=2	0.506	0.517	71.0%	70.3%
soft-BoN	β=0.449	0.964	0.965	22.3%	21.8%
soft-BoN	β=0.232	0.888	0.889	37.5%	38.0%
soft-BoN	β=0.121	0.729	0.725	55.3%	55.4%
soft-BoN	β=0.063	0.506	0.511	70.2%	70.1%
Laplace noise	b=4.42	0.888	0.888	37.3%	37.3%
Laplace noise	b=8.84	0.728		55.0%	
random N	$N \sim \{2, 4, 8\}$	0.715	0.711	41.7%	41.7%

Ranking at L2. Under Gaussian noise the served ranking is the order of the noised scores $\tilde{r}_j$. Under soft-BoN it is drawn without replacement from the softmax (10), one position at a time, which is the Plackett–Luce model of §3. The attacker receives this perturbed ranking at L2 and only $\tilde{y}$ at L1.

E Additional results

E.1 Tournament size and reference count

Two knobs set how much the L1 attacker learns per prompt. The tournament size N belongs to the deployment, and the reference count K belongs to the attacker, who draws K fresh π_{ref} samples per prompt for the L1 objective (6). Table 7 varies each knob on Skywork-RM with one seed. Agreement rises with N, from 0.637 on curated pairs and 0.725 on generated pairs at $N = 2$, to 0.749 and 0.835 at $N = 8$. Agreement grows more slowly with K. $K = 1$ or $K = 3$ reach about 0.72 on curated pairs and 0.79 on generated pairs, against 0.749 and 0.835 with $K = 7$.

E.2 Partial rankings

The weights w_k in ℓ_{L2} (8) let the attacker use only part of the ranking. The winner-only schedule keeps the first term, so the shadow model learns only that the winner beats the other $N-1$ candidates. The top-k schedules keep the first k terms, the top-weighted schedule keeps every term with weight

Table 7: Ordering agreement of the L1 attack on Skywork-RM as the tournament size N and the reference count K vary, one seed each. The $N{=}2$ and $N{=}4$ tournaments are random subsets of the $N{=}8$ candidates, and the $N{=}8$, $K{=}7$ run appears in both groups. Its five-seed mean is $0.754_{\pm 0.051}$ curated and $0.829_{\pm 0.058}$ generated (Table 1).

	tournament size N ($K{=}N{-}1$)			reference count K ($N{=}8$)		
evaluation set	$N{=}2$	$N{=}4$	$N{=}8$	$K{=}1$	$K{=}3$	$K{=}7$
curated	0.637	0.717	0.749	0.716	0.715	0.749
generated	0.725	0.780	0.835	0.788	0.788	0.835

$2^{N-k} - 1$, and the full ranking weights every term equally, which is ℓ_{L2} in the main text. Table 8 reports the sweep. On generated pairs, agreement grows with the part of the ranking the attacker sees, from 0.855 with the winner alone to 0.905 with the full ranking in the base setting, and from 0.942 to 0.979 in the setting of Table 1. On curated pairs the top-2 schedule already reaches the level of the full ranking in the base setting. Even the winner-only schedule is well above L1, since its negatives are the $N-1$ candidates that $r^\star$ ranked below the winner rather than fresh π_{ref} samples.

Table 8: Ordering agreement of the L2 attack when the attacker knows only part of the ranking. The weights w_k of ℓ_{L2} (8) select which positions of the ranking are supervised. The base-setting sweep uses the base Llama-3.1-8B as π_{ref} and as the shadow model's backbone, with one or two seeds per row; its curated set is that of Table 1 and its generated set has 2,349 pairs. In the setting of Table 1 only the winner-only row (one seed) and the full ranking (five seeds) were run.

		base setting			setting of Table 1	
schedule	weights	seeds	curated	generated	curated	generated
winner only	$(1, 0, \ldots, 0)$	2	0.809	0.855	0.882	0.942
top-2	$w_k = 1$ for $k \le 2$	2	0.841	0.875		
top-3	$w_k = 1$ for $k \le 3$	2	0.828	0.881		
top-weighted	$w_k = 2^{N-k} - 1$	1	0.851	0.899		
full ranking	$(1, \ldots, 1)$	1	0.830	0.905	$0.942_{\pm 0.003}$	$0.979_{\pm 0.004}$

E.3 The backbone control

Table 9 repeats the L1 and L2 attacks with a shadow model built on the base Llama-3.1-8B, which does not start from Skywork-RM's own checkpoint.

Table 9: With the base Llama-3.1-8B as its backbone the shadow model loses about 0.02 at both levels. The shadow backbone is the only change, base Llama-3.1-8B in place of Llama-3.1-8B-Instruct, on the same corpus, the same fresh references and the same evaluation sets. The base row is one seed, so it is read against seed 0 of the main-text shadow model.

		L1 `cdf`		L2 `uniform`	
shadow backbone	seeds	curated	generated	curated	generated
Instruct (main text)	5	0.754 ± 0.051	0.829 ± 0.058	0.942 ± 0.003	0.979 ± 0.004
Instruct, seed 0	1	0.749	0.835	0.938	0.974
base, seed 0	1	0.734	0.811	0.920	0.962

E.4 Per-seed values

Table 10 lists every run behind the five-seed means.

Table 10: Ordering agreement of every run behind Table 1 and Table 3, by training seed. Each run is evaluated at its best epoch on the validation set.

RM	evaluation set	level, loss	seed 0	seed 1	seed 2	seed 3	seed 4
Skywork-RM	curated	L1, ℓ_{L1}	0.749	0.739	0.715	0.841	0.724
Skywork-RM	curated	L1, ℓ_{BT}	0.706	0.725	0.655	0.705	0.762
Skywork-RM	curated	L2, ℓ_{L2}	0.938	0.942	0.945	0.943	0.944
Skywork-RM	curated	L3, ℓ_{L3}	0.943	0.943	0.947	0.932	0.944
Skywork-RM	curated	L3, ℓ_{MSE}	0.923	0.936	0.902	0.925	0.933
Skywork-RM	generated	L1, ℓ_{L1}	0.835	0.808	0.775	0.925	0.801
Skywork-RM	generated	L1, ℓ_{BT}	0.777	0.793	0.728	0.776	0.861
Skywork-RM	generated	L2, ℓ_{L2}	0.974	0.985	0.979	0.979	0.979
Skywork-RM	generated	L3, ℓ_{L3}	0.984	0.985	0.980	0.981	0.983
Skywork-RM	generated	L3, ℓ_{MSE}	0.974	0.979	0.956	0.972	0.974
Tulu-RM	generated	L1, ℓ_{L1}	0.769	0.757	0.694	0.758	0.746
Tulu-RM	generated	L1, ℓ_{BT}	0.760	0.750	0.756	0.754	0.757
Tulu-RM	generated	L2, ℓ_{L2}	0.964	0.964	0.963	0.968	0.960
Tulu-RM	generated	L3, ℓ_{L3}	0.966	0.968	0.968	0.971	0.969
Tulu-RM	generated	L3, ℓ_{MSE}	0.962	0.958	0.962	0.963	0.962
RB2-RM	generated	L1, ℓ_{L1}	0.764	0.736	0.757	0.773	0.673
RB2-RM	generated	L1, ℓ_{BT}	0.760	0.754	0.759	0.760	0.758
RB2-RM	generated	L2, ℓ_{L2}	0.972	0.975	0.976	0.969	0.967
RB2-RM	generated	L3, ℓ_{L3}	0.980	0.966	0.976	0.982	0.975
RB2-RM	generated	L3, ℓ_{MSE}	0.968	0.967	0.931	0.955	0.966

E.5 REWARDBENCH-2 BY CATEGORY

Table 11 breaks the accuracy of Table 2 down by category. Two baselines use the base policy itself as the scorer, with no attack. The mean token log-probability of the response under π_{ref} scores 0.417 overall, and the summed log-probability 0.485. The summed variant favors long responses and scores highest on Focus and Ties. The five L1 shadow models of Skywork-RM score 0.240, 0.221, 0.227, 0.498 and 0.236. Under the RewardBench-2 rule, every chosen response must outscore every rejected one, so a random scorer gets 0.238.

Table 11: Accuracy on RewardBench-2 by category. Shadow models are five seeds for every RM (s.d. on the overall accuracy, means elsewhere). A random scorer gets 0.24.

RM	model	overall	Factuality	Precise IF	Math	Safety	Focus	Ties
Skywork-RM	$r^\star$	0.805	0.701	0.412	0.623	0.936	0.956	0.922
	untrained shadow	$0.263_{\pm 0.031}$	0.275	0.265	0.294	0.296	0.261	0.010
	L1 shadow	$0.284_{\pm 0.119}$	0.376	0.236	0.438	0.350	0.132	0.108
	L2 shadow	$0.598_{\pm 0.016}$	0.615	0.358	0.634	0.797	0.438	0.724
	L3 shadow	$0.620_{\pm 0.025}$	0.637	0.351	0.631	0.836	0.466	0.737
Tulu-RM	$r^\star$	0.631	0.724	0.338	0.634	0.736	0.519	0.725
	L1 shadow	$0.249_{\pm 0.003}$	0.370	0.219	0.433	0.244	0.129	0.008
	L2 shadow	$0.543_{\pm 0.010}$	0.684	0.335	0.619	0.675	0.340	0.476
	L3 shadow	$0.557_{\pm 0.009}$	0.688	0.347	0.611	0.625	0.416	0.563
RB2-RM	$r^\star$	0.792	0.743	0.425	0.623	0.891	0.909	0.892
	L1 shadow	$0.239_{\pm 0.006}$	0.337	0.242	0.395	0.232	0.137	0.029
	L2 shadow	$0.624_{\pm 0.020}$	0.659	0.369	0.634	0.777	0.504	0.751
	L3 shadow	$0.633_{\pm 0.047}$	0.658	0.375	0.620	0.757	0.551	0.796

E.6 EVERY DEFENSE CELL

Table 12 lists every defense cell behind Figure 2.

Table 12: Every defense cell, on both evaluation sets. Retained BoN gain ("gain") is the utility arm of App. D (undefended = 1.0). The Gaussian cells at $c \in \{0.5, 1.0\}$ and the soft-BoN cells matched to them are five seeds (mean ± s.d.); every other row is a single seed. Laplace noise (scale b, matched to c) and random N (the winner of a uniformly random subset of $N \in \{2, 4, 8\}$ candidates) are two further mechanisms. The bottom block reports L3 under Gaussian noise.

				L1 `cdf`		L2 `uniform`	
defense	strength	gain	seeds	curated	generated	curated	generated
none (BoN)		1.000	5	0.754 ± 0.051	0.829 ± 0.058	0.942 ± 0.003	0.979 ± 0.004
Gaussian	c=0.25	0.964	1	0.719	0.782	0.937	0.975
Gaussian	c=0.5	0.888	5	0.738 ± 0.023	0.816 ± 0.038	0.922 ± 0.003	0.967 ± 0.002
Gaussian	c=1.0	0.728	5	0.699 ± 0.009	0.760 ± 0.014	0.882 ± 0.007	0.931 ± 0.004
Gaussian	c=2.0	0.506	1	0.690	0.747	0.796	0.867
soft-BoN	β=0.449	0.964	1	0.857	0.924	0.937	0.973
soft-BoN	β=0.232	0.888	5	0.757 ± 0.036	0.835 ± 0.043	0.929 ± 0.005	0.968 ± 0.007
soft-BoN	β=0.121	0.728	5	0.714 ± 0.008	0.768 ± 0.010	0.910 ± 0.003	0.945 ± 0.006
soft-BoN	β=0.063	0.506	1	0.660	0.726	0.858	0.916
Laplace	b=4.42	0.888	1	0.728	0.792	0.925	0.960
Laplace	b=8.84	0.728	1	0.696	0.761	0.896	0.941
random N	$N \sim \{2, 4, 8\}$	0.715	1	0.720	0.773	0.924	0.964
				L3 `pairce`		L3 `logit_mse`	
none (BoN)		1.000	5	0.942 ± 0.006	0.983 ± 0.002	0.924 ± 0.013	0.971 ± 0.009
Gaussian	c=0.5	0.888	1	0.923	0.959	0.919	0.976
Gaussian	c=1.0	0.728	1	0.892	0.921	0.910	0.957