ORDERGRAD: OPTIMIZING BEYOND THE MEAN WITH ORDER-STATISTIC POLICY GRADIENT ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Policy-gradient methods usually optimize expected return, but many real world applications care about distributional properties of returns: tail risk, outlier robustness, or best-of-K discovery. We introduce OrderGrad, a family of likelihood-ratio and reparameterization gradient estimators for order-statistic objectives. OrderGrad optimizes finite-sample L-statistics, i.e., weighted averages of sorted rewards or costs, recovering objectives such as VaR, CVaR, trimmed means, medians, and top-M/best-of-K criteria by changing only the rank weights. For any fixed sample size and rank-weight vector, OrderGrad provides an unbiased gradient estimator for the corresponding order-statistic objective. The method is implemented as a simple reward transformation that can then be used in an otherwise standard policy-gradient or reparameterized update. We study the resulting estimator's variance behavior and evaluate it on tasks where mean optimization is mismatched to the deployment objective, including LLM math post-training and other tasks. OrderGrad provides a unified, plug-and-play route to risk-averse, robust, and exploratory learning.

1 INTRODUCTION

Most stochastic learning algorithms optimize using a gradient estimator $\hat{\boldsymbol{g}}$ that is unbiased for the mean objective $\mathbb{E}[\hat{\boldsymbol{g}}] = \nabla_\theta \mathbb{E}_{x\sim p_\theta}\left[R(x)\right]$. In policy-gradient methods, likelihood-ratio (LR) estimators differentiate log probabilities and weight them by scalar returns or advantages (Williams, 1992); in reparameterized (RP) models, pathwise estimators differentiate samples written as transformations of parameter-free noise (Kingma & Welling, 2014; Rezende et al., 2014). In practice, likelihood-ratio policy gradients have become a workhorse in high-impact applications, including LLM post-training with human or rule-based feedback (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022; Ahmadian et al., 2024; Shao et al., 2024; Guo et al., 2025b) and robot policy learning for motor control and dexterous manipulation (Peters & Schaal, 2006; 2008; OpenAI et al., 2020; 2019). Both viewpoints are now standard in Monte Carlo gradient estimation (Mohamed et al., 2020; Parmas & Sugiyama, 2021). OrderGrad asks how these familiar estimators should change when the objective is not the mean reward but a functional of the whole reward distribution (applicable to both LR and RP as well as regular minibatch gradients).

Why go beyond the mean? The usual criterion $J_{\text{mean}}(\theta) = \mathbb{E}[R]$ compresses the reward distribution to a single number. In robust learning, risk-sensitive RL, evaluation of stochastic agents, and preference optimization, this mean can be the wrong target. An inference-time generation system may sample several completions and select one using a reward model (Stiennon et al., 2020; Nakano et al., 2021); a safety-critical controller may care about lower-tail or percentile risk (Chow et al., 2015; 2018); a robust learner may trim high-loss corrupted samples (Shen & Sanghavi, 2019; Lugosi & Mendelson, 2021); and an alignment or preference-learning pipeline may target specific reward quantiles or top-M rankings rather than the mean (Wang et al., 2025b; Cai et al., 2025). Order statistics are a standard way to describe such distributional goals: they expose lower tails, upper tails, medians, trimmed means, and best-of-k behavior using only sorted outcomes. More generally, weighted averages of order statistics, or L-statistics, provide a classical and flexible language for robust and distribution-aware objectives (Daniell, 1920; Bickel & Lehmann, 1975; Huber & Ronchetti, 2009; Maurer et al., 2021).

Order statistics as quantile weighting. OrderGrad starts from a simple idea: instead of summarizing a reward distribution only by its mean, we sort a small group of sampled rewards and decide which

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.

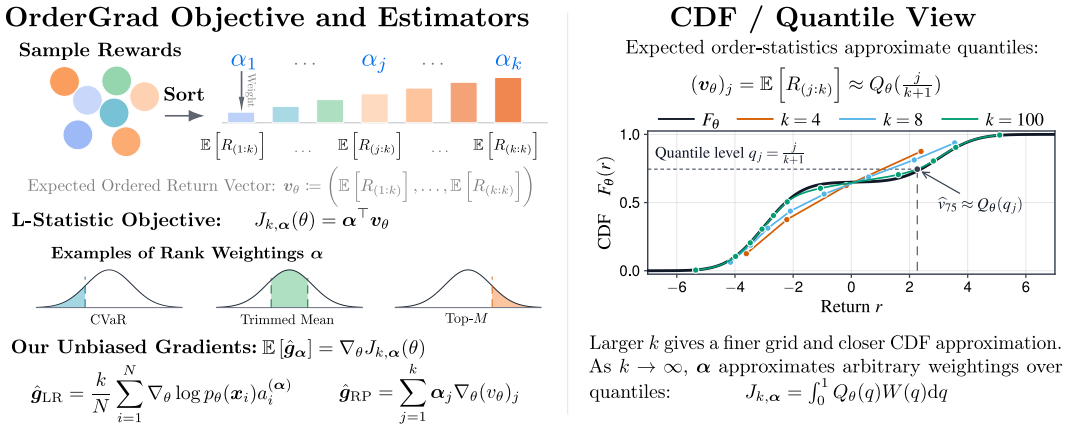


Figure 1: OrderGrad overview. Rank weights define a distributional objective over sorted rewards. For likelihood-ratio gradients, OrderGrad transforms rewards into rank advantages $\{R_i\} \mapsto \{a_i^{(\alpha)}\}$; for reparameterization gradients, it differentiates the rank-weighted value v^α. The right side shows the equivalent quantile view of the finite-k objective, shown for a bimodal reward distribution $p(r)$.

ranks matter. Given k samples, write their sorted rewards as

$$R_{(1:k)} \le \cdots \le R_{(k:k)}.$$

The lowest ranks describe the lower tail, the middle ranks describe typical outcomes, and the highest ranks describe best-of-k behavior. As k grows, these sorted rewards give an increasingly fine approximation to the reward distribution: the jth order statistic corresponds roughly to the $j/(k+1)$ quantile, and in the limit this view recovers the underlying CDF/quantile curve (Fig. 1, right).

Thus, choosing rank weights $\alpha_1, \ldots, \alpha_k$ is a way of choosing which parts of the reward distribution to optimize. Placing weight on large ranks emphasizes high-quantile or best-of-k performance; placing weight on small ranks emphasizes robustness or safety in the lower tail; spreading weight over a range gives trimmed or tail-averaged objectives. The resulting objective is

$$J_{k,\alpha}(\theta) \coloneqq \mathbb{E}\left[\sum_{j=1}^{k} \alpha_j R_{(j:k)}\right] = \sum_{j=1}^{k} \alpha_j \mathbb{E}\left[R_{(j:k)}\right]. \tag{1}$$

OrderGrad as a simple transformation. OrderGrad derives unbiased estimators for (1) by changing only the scalar learning signal used by familiar gradient estimators. In likelihood-ratio form, the minibatch rewards are transformed into rank advantages, with one line of code,

$$\{R_i\}_{i=1}^N \longmapsto \{a_i^{(\alpha)}\}_{i=1}^N, \qquad \texttt{a = ordergrad.lstat_advantage(rewards, alpha)}$$

which are then plugged into an otherwise standard policy-gradient update. The reparameterization form has a similar transform. The explicit LR and RP estimator equations are summarized in Fig. 1 and written in (11); the appendix gives the unbiasedness derivations. The order-statistic size k defines the target criterion and can be smaller than the batch size N. For fixed N, k, and α, the required rank weights can be precomputed and collapsed once, so each minibatch update is sorting plus a linear pass, with total time $O(N \log N)$ dominated by the sort (i.e., negligible in practice).

From the technical perspective, the closest works are the recent line of approaches for unbiased pass@k/Max@k optimization (Koyamada et al., 2023b; Nishimori et al., 2026; Tang et al., 2025; Walder & Karkhanis, 2025; Chen et al., 2025b; Bagirov et al., 2025), which are a special case of order-statistics (specifically, they consider only the top rank). In this line of work, the key question has been how to reduce gradient estimation variance. Koyamada et al. (2023b) initially gave a rudimentary LR based estimation method with a baseline computed from separate batches; Tang et al. (2025) proposed a better baseline that uses the max of the other $k-1$ samples; Walder & Karkhanis (2025) extended this approach to cases when $N > k$ to achieve better variance reduction; and Chen et al. (2025b) and Bagirov et al. (2025) give yet other baselines based on mean estimates

for pass@k and Max@k respectively. Our work could be seen as a generalization of these methods to arbitrary order-statistics beyond only the top-1 rank. We give a different derivation for the baseline computation that allows generalization to our scenario, but matches MaxPO (Takashiro et al., 2026) for the top-1 rank (Max@k). The centered on-policy estimators of MaxRL (Tajwar et al., 2026) and TailRL (Ramasubramanian et al., 2026) are also special cases of OrderGrad (Appendix H.3).

Experimentally, we evaluate OrderGrad on LLM math post-training tasks in Sec. 5. We use Top-M@K objectives, which average the top M members of each K-sample group, and show improved pass@k compared with the Max@K special case. We also combine a Top-M@K solve reward with a Bottom-M@K length-penalty cost. This targeted rank weighting significantly shortens outputs without substantially reducing solve rate, whereas a naive GRPO scalarization that simply sums the solve and length rewards gives poor performance.

2 Preliminaries

2.1 Notation and Monte Carlo gradient estimators

Let $\theta \in \mathbb{R}^d$ denote the parameters of a sampler, policy, model, or stochastic computation graph. A random object $x \sim p_\theta$ may be a trajectory in RL, a completion from a language model, an action in a bandit, or a latent variable in a differentiable model. A scalar reward is denoted by $R(x) \in \mathbb{R}$; larger values are better. For loss minimization, we set $R = -L$.

For a generic scalar learning criterion $J(\theta)$, write

$$\boldsymbol{g}(\theta) \coloneqq \nabla_\theta J(\theta). \tag{2}$$

Following common Monte Carlo gradient-estimation notation, an estimator $\hat{\boldsymbol{g}}$ is a random vector. It is unbiased for J when

$$\mathbb{E}[\hat{\boldsymbol{g}}] = \boldsymbol{g}(\theta) = \nabla_\theta J(\theta). \tag{3}$$

With N independent samples, the usual batch estimator is $\hat{\boldsymbol{g}}_N = N^{-1} \sum_{i=1}^N \hat{\boldsymbol{g}}_i$.

2.2 The mean objective and elementary estimators

The standard expected-reward objective is

$$J_{\text{mean}}(\theta) = \mathbb{E}_{x \sim p_\theta}[R(x)]. \tag{4}$$

The likelihood-ratio, or score-function, estimator is

$$\hat{\boldsymbol{g}}_{\text{LR}} = \nabla_\theta \log p_\theta(x)\big(R(x) - b\big), \qquad \mathbb{E}[\hat{\boldsymbol{g}}_{\text{LR}}] = \nabla_\theta J_{\text{mean}}(\theta), \tag{5}$$

where b is a baseline independent of the current sample x; more generally, b may depend on the state, prompt, context, or other samples, but not on the sampled action or trajectory whose score is being multiplied. Under this condition $\mathbb{E}[\nabla_\theta \log p_\theta(x) b] = 0$, so the estimator remains unbiased. This includes population baselines and leave-one-out baselines that exclude the current sample, as in multi-sample Monte Carlo estimators (Mnih & Rezende, 2016; Parmas et al., 2018).

When the sample can be written as a differentiable transformation $x = T_\theta(\varepsilon)$ with $\varepsilon \sim p(\varepsilon)$ independent of θ, the reparameterization/pathwise estimator is

$$\hat{\boldsymbol{g}}_{\text{RP}} = \nabla_\theta R\big(T_\theta(\varepsilon)\big) = \nabla_x R(x)\, \nabla_\theta T_\theta(\varepsilon), \tag{6}$$

assuming R has no additional direct dependence on θ. LR estimators require only rewards and log-probability gradients, while RP estimators can have lower variance but require a differentiable sampling path. OrderGrad can be layered on top of either estimator family: in LR form it replaces $R - b$ by a rank-based batch advantage; in RP form it differentiates the batch value v^α.

2.3 Order statistics and L-statistics

To move beyond the mean objective, OrderGrad draws k rewards, sorts them as $R_{(1:k)} \le \cdots \le R_{(k:k)}$, and optimizes the rank-weighted criterion $J_{k,\alpha}(\theta) = \mathbb{E}\Big[\sum_{j=1}^k \alpha_j R_{(j:k)}\Big]$. An L-statistic is a linear

combination of order statistics. In this paper, the order-statistic sample size k is a design choice that determines which distributional property the gradient estimator targets. The weight vector α can be normalized to sum to one, but this is not required: unnormalized weights are useful when scaling gradients or combining multiple criteria.

A simple finite-population computation will be used repeatedly below. Suppose a fixed batch has sorted rewards

$$R_{(1:N)} \le \cdots \le R_{(N:N)}$$

and let S be a uniformly random size-k subset of the N indices. The probability that the mth sorted batch reward becomes the jth order statistic of the subset is

$$\omega_{m,j}^{(N,k)} = \mathbb{P}\Big((R_S)_{(j:k)} = R_{(m:N)} \mid R_{1:N}\Big) = \frac{\binom{m-1}{j-1}\binom{N-m}{k-j}}{\binom{N}{k}}, \tag{7}$$

with the convention that out-of-range binomial coefficients are zero. This is the standard order-statistic distribution for a simple random sample without replacement from a finite population (Arnold et al., 1992, Sec. 3.7); see also O'Neill (2025). Consequently the batch value of the jth order statistic can be computed as the fixed weighted sum

$$v_j = \mathbb{E}\Big[(R_S)_{(j:k)} \mid R_{1:N}\Big] = \sum_{m=1}^{N} R_{(m:N)}\, \omega_{m,j}^{(N,k)}. \tag{8}$$

This counting idea is the basis for the OrderGrad estimator; below we extend it to obtain the include-one value and a principled leave-one-out baseline for the LR gradient estimator.

2.4 Connection to CDFs, quantiles, and spectral criteria

For a continuous reward distribution with CDF F and quantile function $Q(u) = F^{-1}(u)$, the probability integral transform lets us sample rewards by first drawing $U_i \sim \mathrm{Unif}(0,1)$ and then setting $R_i \coloneqq Q(U_i)$. Since Q is nondecreasing, ranking can be carried out in the uniform sample space and then mapped back through Q:

$$U_{(1:k)} \le \cdots \le U_{(k:k)}, \qquad R_{(j:k)} \coloneqq Q(U_{(j:k)}).$$

The jth order statistic of k independent uniforms has distribution $U_{(j:k)} \sim \mathrm{Beta}(j, k+1-j)$ (Arnold et al., 1992); write its density as $b_{j,k}$. This Beta density is a smooth kernel on $[0,1]$ with mean $j/(k+1)$. Therefore

$$\mathbb{E}[R_{(j:k)}] = \int_0^1 Q(u) b_{j,k}(u)\, du, \qquad J_{k,\alpha}(\theta) = \int_0^1 Q_\theta(u) w_{k,\alpha}(u)\, du, \tag{9}$$

where $w_{k,\alpha}(u) = \sum_{j=1}^{k} \alpha_j b_{j,k}(u)$. Thus finite-$k$ order objectives are beta-kernel-smoothed quantile-weighted objectives, with α mixing Beta kernels over quantile levels. As k grows, $J_{k,\alpha}$ approaches arbitrary quantile-weighted criteria $\int_0^1 Q_\theta(u) W(u)\, du$, including CVaR/AVaR and spectral risk measures (Acerbi & Tasche, 2002b;a; Shapiro, 2013; Holland & Haress, 2022).

3 OrderGrad method: Unbiased Order-statistic Gradient Estimation

3.1 From population advantages to batch advantages

We first derive the likelihood-ratio form in the ideal population setting. Draw

$$A_1, \ldots, A_k \overset{i.i.d.}{\sim} p_\theta, \qquad R_\ell = R(A_\ell), \qquad R_{(1:k)} \le \cdots \le R_{(k:k)}.$$

For a single rank j, define the population order-statistic value and the conditional value of fixing the first draw to action b:

$$\bar{v}_j(\theta) \coloneqq \mathbb{E}[R_{(j:k)}], \qquad \bar{q}_{b,j} \coloneqq \mathbb{E}[R_{(j:k)} \mid A_1 = b].$$

Analogously to Nishimori et al. (2026), applying the REINFORCE identity to the k i.i.d. draws and using exchangeability gives

$$\nabla_\theta \bar{v}_j(\theta) = \mathbb{E}\left[R_{(j:k)} \sum_{\ell=1}^{k} \nabla_\theta \log p_\theta(A_\ell)\right] = k\,\mathbb{E}\Big[\nabla_\theta \log p_\theta(A_1) R_{(j:k)}\Big]$$
$$= k\,\mathbb{E}_{b\sim p_\theta}\Big[\nabla_\theta \log p_\theta(b)\,\mathbb{E}[R_{(j:k)} \mid A_1 = b]\Big] = k\,\mathbb{E}_{b\sim p_\theta}\big[\nabla_\theta \log p_\theta(b)\,\bar{q}_{b,j}\big]. \tag{10}$$

The unconditional value $\bar{v}_j = \mathbb{E}_{b\sim p_\theta}[\bar{q}_{b,j}]$ is a natural baseline for the conditional value. In the sampling estimator below, we use this idea through a leave-one-out baseline that does not require access to $\bar{v}_j$. For L-statistic weights $\alpha \in \mathbb{R}^k$, define

$$\bar{v}^\alpha := \sum_{j=1}^{k} \alpha_j \bar{v}_j, \qquad \bar{q}_b^{(\alpha)} := \sum_{j=1}^{k} \alpha_j \bar{q}_{b,j}.$$

When the distribution and reward table are known, the barred values and gradients can be computed analytically; the binomial-tail formulas are given in App. C. In ordinary minibatch optimization, however, we do not have access to $\bar{q}_{b,j}$ or $\bar{v}_j$. The rest of the main method therefore constructs sampling-based analogues from a realized batch: the value v_j, the include-one value $q_{i,j}$, the leave-one-out baseline $v_j^{(-i)}$, and the advantage $a_{i,j} = q_{i,j} - v_j^{(-i)}$.

The next subsections derive these batch quantities and then plug them into the two unbiased sample-based estimators

$$\hat{\boldsymbol{g}}^{\alpha}_{\text{LR-OG}} = \frac{k}{N} \sum_{i=1}^{N} a_i^{(\alpha)}\, \nabla_\theta \log p_\theta(x_i), \qquad \hat{\boldsymbol{g}}^{\alpha}_{\text{RP-OG}} = \nabla_\theta v^\alpha(\theta) = \nabla_\theta \sum_{j=1}^{k} \alpha_j v_j(\theta). \tag{11}$$

The LR estimator uses the leave-one-out-centered batch advantage. A standard interchange of differentiation and expectation gives the RP route: estimate the rank-weighted value by v^α on the batch and differentiate it directly.

3.2 Batch values, include-one values, and advantages

This subsection defines the sample-based estimates used in (11). Fix a realized batch of rewards $R_{1:N}$. For the value estimator we require $1 \le k \le N$; for the LR leave-one-out advantage below we require $1 \le k < N$, so that a size-k subset can still be drawn after removing one datapoint. For any size-k subset $S \subseteq [N]$, let $(R_S)_{(j:k)}$ denote the jth smallest reward in that subset. The batch value for rank j is the average over all such subsets,

$$v_j := \frac{1}{\binom{N}{k}} \sum_{\substack{S\subseteq[N]\\ |S|=k}} (R_S)_{(j:k)}. \tag{12}$$

For datapoint i, the include-one and leave-one-out values average over all size-k subsets that contain or exclude i, respectively:

$$q_{i,j} := \frac{1}{\binom{N-1}{k-1}} \sum_{\substack{S\subseteq[N]\\ |S|=k,\ i\in S}} (R_S)_{(j:k)}, \tag{13}$$

$$v_j^{(-i)} := \frac{1}{\binom{N-1}{k}} \sum_{\substack{S\subseteq[N]\setminus\{i\}\\ |S|=k}} (R_S)_{(j:k)}. \tag{14}$$

The batch advantage is

$$a_{i,j} := q_{i,j} - v_j^{(-i)}, \qquad a_i^{(\alpha)} := \sum_{j=1}^{k} \alpha_j a_{i,j}. \tag{15}$$

These subset averages are the unbarred analogues of the barred population quantities. For an i.i.d. batch, they preserve unbiasedness by iterated expectation. The leave-one-out baseline excludes sample i, so it is independent of the score term, matching the condition in Sec. 2.2. This extends the baseline construction of Takashiro et al. (2026): for the maximum, $q_{i,k} - v_k^{(-i)}$ is exactly their canonical marginal advantage. Counting subset incidences also gives $\sum_i a_i^{(\alpha)} = 0$ for every batch.

3.3 EXPLICIT BATCH EQUATIONS AND FAST COMPUTATION

This subsection gives efficient computation for the sample-based quantities defined above. All batch quantities can be computed after a single sort. Let

$$R_{(1:N)} \le \cdots \le R_{(N:N)}$$

be the sorted rewards, and map the resulting quantities back to the original datapoint indices by the inverse sorting permutation.

For the U-statistic value, the probability that sorted reward $R_{(m:N)}$ is the jth order statistic of a uniform size-k subset is

$$W_{m,j} = \frac{\binom{m-1}{j-1}\binom{N-m}{k-j}}{\binom{N}{k}}, \qquad m = 1, \dots, N, \quad j = 1, \dots, k. \tag{16}$$

Therefore

$$v_j = \sum_{m=1}^{N} R_{(m:N)} W_{m,j}, \qquad v^\alpha = \sum_{m=1}^{N} R_{(m:N)} w_m^{(\alpha)}, \qquad w^{(\alpha)} = W\alpha. \tag{17}$$

For include-one values, let t denote the sorted rank of the datapoint forced into the subset. Define precomputable weight tables

$$A_{m,j} \coloneqq \frac{\binom{m-1}{j-1}\binom{N-m-1}{k-j-1}}{\binom{N-1}{k-1}}, \quad B_{m,j} \coloneqq \frac{\binom{m-1}{j-1}\binom{N-m}{k-j}}{\binom{N-1}{k-1}}, \quad C_{m,j} \coloneqq \frac{\binom{m-2}{j-2}\binom{N-m}{k-j}}{\binom{N-1}{k-1}}. \tag{18}$$

Then

$$q_{t,j} = \sum_{m<t} R_{(m:N)} A_{m,j} + R_{(t:N)} B_{t,j} + \sum_{m>t} R_{(m:N)} C_{m,j}. \tag{19}$$

For leave-one-out values, let

$$W^-_{\ell,j} \coloneqq \frac{\binom{\ell-1}{j-1}\binom{(N-1)-\ell}{k-j}}{\binom{N-1}{k}}, \qquad \ell = 1, \dots, N-1. \tag{20}$$

After removing sorted rank t,

$$v_j^{(-t)} = \sum_{m<t} R_{(m:N)} W^-_{m,j} + \sum_{m>t} R_{(m:N)} W^-_{m-1,j}. \tag{21}$$

Finally,

$$a_{t,j} = q_{t,j} - v_j^{(-t)}, \qquad a_t^{(\alpha)} = \sum_{j=1}^{k} \alpha_j a_{t,j}. \tag{22}$$

The appendix derives (16)–(21) by counting subsets and gives prefix/suffix formulas. Rankwise computation costs $O(Nk)$ after sorting. For fixed α, collapsed weights give all $a_t^{(\alpha)}$ in $O(N)$ after sorting, so the total minibatch cost is $O(N \log N)$.

4 SMALL DIAGNOSTIC EXPERIMENTS

We first run small diagnostic experiments that isolate the two basic claims needed before the larger task evaluations: the batch quantities are cheap to compute, and the LR/RP estimators recover the intended order-statistic gradients.

Weight visualization and runtime. The first diagnostic studies the computation of the batch quantities from Sec. 3.3. The left panel of Fig. 2 visualizes the weights placed on sorted batch ranks when computing v^α for several choices of α. The middle panel measures the runtime of `lstat_advantage` as the batch size N grows. The right panel fixes $N = 5000$ and varies k, comparing the collapsed `lstat_advantage` computation against the full rankwise order-statistic computation. The results show that the extra OrderGrad computation is negligible: even for batch sizes around 10^4, the update takes about 1 ms, so we do not expect it to be a bottleneck in practice.

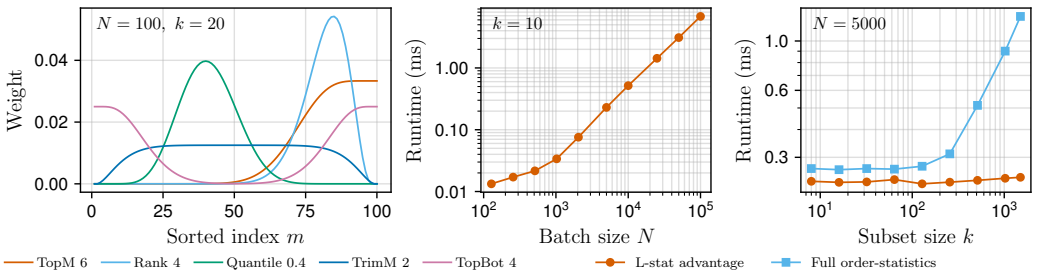


Figure 2: Diagnostic computation experiment. The panels visualize several rank-weight choices for v^{α} and measure the runtime of the practical collapsed OrderGrad computation.

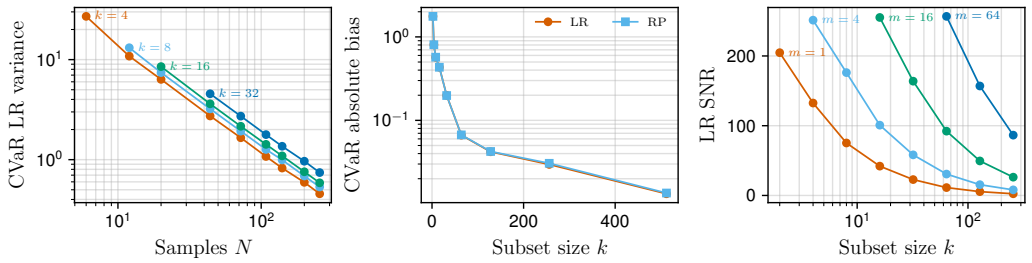


Figure 3: Diagnostic gradient experiments. For $x \sim \mathcal{N}(\mu = 0.5, 1)$ and $R(x) = -x^2$, we compare LR and RP estimates to the exact 20% CVaR gradient and study the SNR of Top-M@K estimators.

Gradient bias, variance, and SNR. The second diagnostic checks gradient estimation in a one-dimensional setting where the target gradient can be computed accurately. We use $x \sim \mathcal{N}(\mu, 1)$ with reward $R(x) = -x^2$, compute the exact gradient of the 20% CVaR objective with respect to μ, and compare repeated LR and RP estimates across different values of k and N. The left panel of Fig. 3 reports the LR estimator variance as a function of N for several choices of k. The middle panel reports empirical bias against k, comparing the LR and RP estimates to the exact gradient. Increasing k increases variance, but it also reduces bias relative to the exact CVaR gradient. Thus larger k approximates the target objective more accurately, at the cost of higher estimation variance.

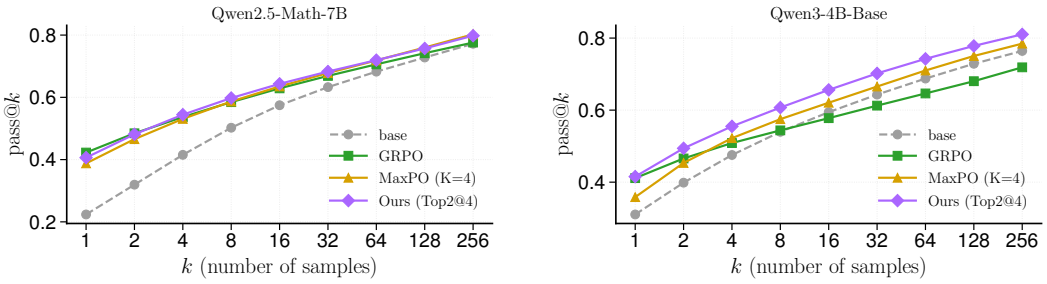
Top-M@K SNR. The right panel of Fig. 3 studies the signal-to-noise ratio of the Top-M@K estimator as M varies. This experiment isolates the effect of smoothing the sharp Max@K objective: increasing M averages over more high-ranked samples, which should reduce estimator noise while retaining pressure on the upper tail.

5 LLM Reasoning Experiments

We evaluate our Top-M@K method on challenging math reasoning tasks, asking: (i) does it improve pass@k over GRPO and MaxPO (Max@K Policy Optimization; Takashiro et al., 2026), whose centered advantage is the maximum-rank special case of OrderGrad; (ii) how does M affect performance at fixed K; and (iii) how does K affect performance at fixed M.

5.1 Experimental Setting

Training. We perform RL fine-tuning on Qwen3-4B-Base (Yang et al., 2025) and Qwen2.5-Math-7B (Yang et al., 2024). Our baselines are GRPO (Shao et al., 2024) and MaxPO (Takashiro et al., 2026). MaxPO is the $M = 1$ special case of our top-M method. By default, we use Top-2@4 with $K = 4$; see Sec. 5.3 for a full sensitivity analysis on M and K and App. J.1 for training data and hyperparameters. Here, K is the training comparison size (theoretical k), and M counts the top-ranked rewards averaged in our objective. Evaluation uses pass@k.



(a) Qwen2.5-Math-7B (b) Qwen3-4B-Base

Figure 4: Task-average pass@k ($k \leq 256$). Unweighted average over AIME24, AIME25, AMC23, MATH500, and Minerva (temperature 0.6, top-p 0.95, $n = 1024$ per problem). Our method with Top-2@4 outperforms GRPO at large k and outperforms MaxPO ($K = 4$) overall on pass@k.

Evaluation. We use five math reasoning benchmarks: AIME24, AIME25, AMC23, MATH500 (Hendrycks et al., 2021), and Minerva (Lewkowycz et al., 2022). We use nucleus sampling (temperature 0.6, top-p 0.95), generating $n = 1024$ samples per problem to reduce evaluation variance. We report unbiased pass@k (Chen et al., 2021) for $k \in \{1, 2, 4, 8, \ldots\}$:

$$\text{pass@}k \;:=\; \mathbb{E}_{x\sim\mathcal{D}}\left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right], \tag{23}$$

where n is the number of sampled completions and c is the number of correct completions among them. We report results up to $k \leq 256$ for all benchmarks.

5.2 Results

Figure 4 shows task-average pass@k curves. Against GRPO, OrderGrad improves pass@256 by 0.092 on Qwen3-4B-Base and 0.022 on Qwen2.5-Math-7B, and it avoids GRPO's diversity collapse on Qwen3-4B-Base. Against MaxPO ($K = 4$), Top-2@4 improves most of the pass@k profile: pass@1 improves by 0.057 on Qwen3-4B-Base and 0.018 on Qwen2.5-Math-7B, and pass@256 improves by 0.026 on Qwen3-4B-Base, while the largest-k Qwen2.5-Math-7B results remain comparable to MaxPO ($K = 4$). Per-benchmark curves and pass@1/pass@256 are in App. J.2 (Figures 15–16 and Table 6).

5.3 Effective Size of M, K

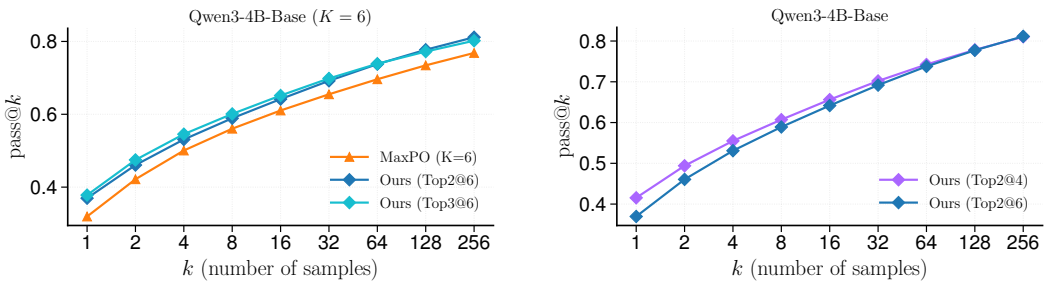
We analyze the sensitivity of our method to the two hyperparameters on Qwen3-4B-Base (Figure 5): the parameter M at fixed $K = 6$, and the training objective size K at fixed $M = 2$. Corresponding results on Qwen2.5-Math-7B are reported in App. J.2 (Tables 4 and 6).

Effect of M ($K = 6$ fixed). We compare $M \in \{2, 3\}$ against MaxPO ($K = 6$) (Figure 5a). Both configurations improve pass@1 by more than 0.05 and pass@256 by 0.034–0.043 over MaxPO ($K = 6$) on Qwen3-4B-Base. Our baseline averages the top-M rewards of the leave-one-out group; larger M averages over more samples and yields a better gradient estimate.

Effect of K ($M = 2$ fixed). We compare $K \in \{4, 6\}$ (Figure 5b). Increasing K from 4 to 6 decreases pass@1 while increasing pass@256. This indicates an exploration–exploitation trade-off: increasing K promotes broader exploration, which reduces one-shot accuracy but improves the probability of success when multiple samples are available.

5.4 Different Reward Objectives

We use a correctness reward for the top M rollouts and a response-length reward for the bottom M rollouts (shorter responses receive higher reward). In RL for LLMs, such as GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025), a common approach applies a weighted sum of correctness and length rewards uniformly to all rollouts. However, interference from the length penalty can lead



(a) Effect of M at fixed $K = 6$.

(b) Effect of K at fixed $M = 2$.

Figure 5: Effective size of M and K on Qwen3-4B-Base.

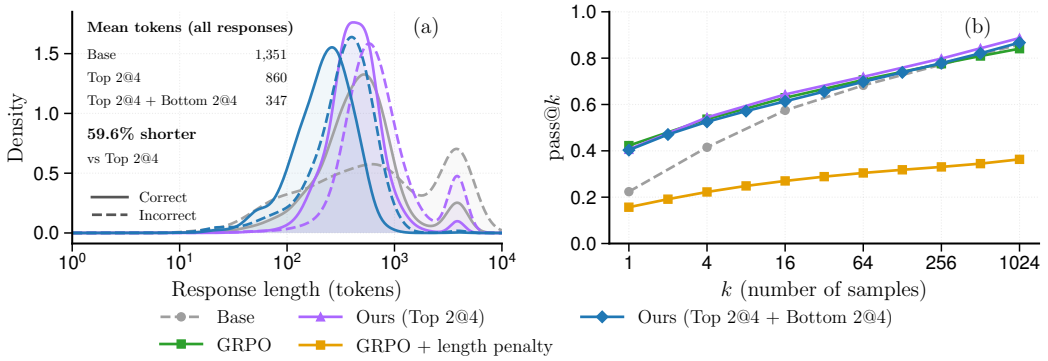


Figure 6: Results for Multi-Reward Objectives with Correctness Reward and Length Penalty on Qwen2.5-Math-7B. (a) Minerva response lengths (solid: correct; dashed: incorrect); inset means pool all responses, with reduction relative to Top-2@4. (b) Task-average pass@k ($k \leq 1024$), averaged uniformly over AIME24, AIME25, AMC23, MATH500, and Minerva. Colors identify the same models in both panels (temperature 0.6, top-p 0.95, $n = 1024$ per problem).

to degenerate short outputs. Moreover, LLMs tend to overthink (Sui et al., 2025) when producing incorrect responses (Fig. 6a), and the longest rollout can bottleneck Best-of-N inference. We compare two objectives. The GRPO baseline uses $r = r_{\text{correct}} - 0.2 \cdot L/L_{\max}$, where correctness reward $r_{\text{correct}} \in \{0, 1\}$, L denotes the response length, and $L_{\max} = 3072$. OrderGrad combines Top-2@4 and Bottom-2@4 ($M = 2$, $K = 4$). The top two rollouts use a correctness weight of 1.0, while the two longest rollouts use a length-penalty weight of $0.4 \cdot L/L_{\max}$. As shown in Fig. 6b, the length penalty harms GRPO, leading to poor performance and output collapse. In contrast, OrderGrad preserves pass@k performance close to the $K = 4$, $M = 2$ setting while greatly reducing response length; in particular, extremely long responses are eliminated (Fig. 6a).

6 Discussion, Limitations and Conclusions

OrderGrad exposes a large design space: choosing k and α determines where gradient pressure falls across the reward distribution. Smaller k and smoother weights can stabilize learning, while larger k and sharper weights better match extreme deployment metrics but may increase variance. We explored only a focused subset over k, α, reward definitions, and task domains. Practical use also requires careful consistent tie/atom handling, and accounting for off-policy data, reward-model misspecification, prompt dependence, and PPO clipping/normalization in LLM settings.

OrderGrad is a powerful method that unifies robust, safe, and exploratory metrics under one rank-weighted gradient-estimation framework. The method is flexible and easy to use because it can be implemented as a reward or advantage transformation and plugged into standard LR or RP updates. We hope the community finds OrderGrad useful for optimizing beyond the mean.

AI USE STATEMENT

We used large language models (LLMs) to assist with prose editing, literature review, coding, experiments, and proofs. The research idea originated with the authors, who used iterative, back-and-forth prompting to implement that idea with LLM assistance. We take responsibility for the final manuscript, including its claims, proofs, code, experimental results, and citations.

REFERENCES

Carlo Acerbi. Spectral measures of risk: A coherent representation of subjective risk aversion. *Journal of Banking & Finance*, 26(7):1505–1518, 2002.

Carlo Acerbi and Dirk Tasche. On the coherence of expected shortfall. *Journal of Banking & Finance*, 26(7):1487–1503, 2002a.

Carlo Acerbi and Dirk Tasche. Expected shortfall: A natural coherent alternative to value at risk. *Economic Notes*, 31(2):379–388, 2002b.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 12248–12267, 2024. doi: 10.18653/v1/2024.acl-long.662.

Barry C. Arnold, N. Balakrishnan, and H. N. Nagaraja. *A First Course in Order Statistics*. John Wiley & Sons, 1992.

Farid Bagirov, Mikhail Arkhipov, Ksenia Sycheva, Evgeniy Glukhov, and Egor Bogomolov. The best of N worlds: Aligning reinforcement learning with best-of-N sampling via max@k optimisation. *arXiv preprint arXiv:2510.23393*, 2025.

Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, T. B. Dhruva, Alistair Muldal, Nicolas Heess, and Timothy P. Lillicrap. Distributed distributional deterministic policy gradients. In *International Conference on Learning Representations*, 2018.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 449–458. PMLR, 2017.

Marc G. Bellemare, Will Dabney, and Mark Rowland. *Distributional Reinforcement Learning*. The MIT Press, Cambridge, MA, 2023.

Peter J. Bickel and Erich L. Lehmann. Descriptive statistics for nonparametric models. II. location. *The Annals of Statistics*, 3(5):1045–1069, 1975.

Dake Bu, Wei Huang, Andi Han, Atsushi Nitanda, Bo Xue, Qingfu Zhang, Hau-San Wong, and Taiji Suzuki. Consistency is not always correct: Towards understanding the role of exploration in post-training reasoning. *arXiv preprint arXiv:2511.07368*, 2025.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Shihao Cai, Chongming Gao, Yang Zhang, Wentao Shi, Jizhi Zhang, Keqin Bao, Qifan Wang, and Fuli Feng. K-order ranking preference optimization for large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4844–4859, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.250. URL https://aclanthology.org/2025.findings-acl.250/.

Adrian Rivera Cardoso and Huan Xu. Risk-averse stochastic convex bandit. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 39–47, 2019.

Fan Chen, Audrey Huang, Noah Golowich, Sadhika Malladi, Adam Block, Jordan T. Ash, Akshay Krishnamurthy, and Dylan J. Foster. The coverage principle: How pre-training enables post-training. *arXiv preprint arXiv:2510.15020*, 2025a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025b.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*, 2025.

Taehyun Cho, Seungyub Han, Seokhun Ju, Dohyeong Kim, Kyungjae Lee, and Jungwoo Lee. Bellman unbiasedness: Toward provably efficient distributional reinforcement learning with general value function approximation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10499–10523. PMLR, 2025.

Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: A CVaR optimization approach. In *Advances in Neural Information Processing Systems*, volume 28, pp. 1522–1530, 2015.

Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18 (167):1–51, 2018. URL https://jmlr.org/papers/v18/15-636.html.

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*, volume 30, 2017.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Sebastian Curi, Kfir Y. Levy, Stefanie Jegelka, and Andreas Krause. Adaptive sampling for stochastic risk-averse learning. In *Advances in Neural Information Processing Systems 33*, pp. 1036–1047, 2020.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1096–1105. PMLR, 2018a.

Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 2892–2901. AAAI Press, 2018b.

Xingyu Dang, Christina Baek, Kaiyue Wen, Zico Kolter, and Aditi Raghunathan. Weight ensembling improves reasoning in language models. In *Second Conference on Language Modeling*, 2025.

P. J. Daniell. Observations weighted according to order. *American Journal of Mathematics*, 42(4): 222–236, 1920.

Perry Dong, Chongyi Zheng, Chelsea Finn, Dorsa Sadigh, and Benjamin Eysenbach. Value flows. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2VyNYUVF2k.

Yanbo Fan, Siwei Lyu, Yiming Ying, and Baogang Hu. Learning with average top-k loss. In *Advances in Neural Information Processing Systems 30*, 2017.

Jingtong Gao, Ling Pan, Yejing Wang, Rui Zhong, Chi Lu, Qingpeng Cai, Peng Jiang, and Xiangyu Zhao. Navigate the unknown: Enhancing LLM reasoning with intrinsic motivation guided exploration. *arXiv preprint arXiv:2505.17621*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025a.

Daya Guo et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025b. doi: 10.1038/s41586-025-09422-z.

Andre Wang He, Daniel Fried, and Sean Welleck. Rewarding the unlikely: Lifting GRPO beyond distribution sharpening. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25548–25560, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 3215–3222. AAAI Press, 2018.

Matthew J. Holland and El Mehdi Haress. Learning with risk-averse feedback under potentially heavy tails. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 892–900, 2021.

Matthew J. Holland and El Mehdi Haress. Spectral risk-based learning using unbounded losses. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 1871–1886, 2022.

Matthew J. Holland and Kazuki Tanabe. A survey of learning criteria going beyond the usual risk. *Journal of Artificial Intelligence Research*, 78:781–821, 2023.

Sihan Hu, Xiansheng Cai, Yuan Huang, Zhiyuan Yao, Linfeng Zhang, Pan Zhang, Youjin Deng, and Kun Chen. Emergent slow thinking in LLMs as inverse tree freezing. *arXiv preprint arXiv:2509.23629*, 2025.

Peter J. Huber and Elvezio M. Ronchetti. *Robust Statistics*. John Wiley & Sons, 2 edition, 2009.

Yash Jhaveri, Harley Wiltzer, Patrick Shafto, Marc Bellemare, and David Meger. Convergence theorems for entropy-regularized and distributional reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Yuxian Jiang, Yafu Li, Guanxu Chen, Dongrui Liu, Yu Cheng, and Jing Shao. Rethinking entropy regularization in large reasoning models. *arXiv preprint arXiv:2509.25133*, 2025.

Tyler Kastner, Mark Rowland, Yunhao Tang, Murat A. Erdogdu, and Amir-Massoud Farahmand. Categorical distributional reinforcement learning with kullback-leibler divergence: Convergence and asymptotics. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 29294–29320. PMLR, 2025.

Justin Khim, Liu Leqi, Adarsh Prasad, and Pradeep Ravikumar. Uniform convergence of rank-weighted learning. In *International conference on machine learning*, pp. 5254–5263. PMLR, 2020.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Sotetsu Koyamada, Shinri Okano, Soichiro Nishimori, Yu Murata, Keigo Habara, Haruka Kita, and Shin Ishii. pgx: Hardware-accelerated parallel game simulators for reinforcement learning. *Advances in Neural Information Processing Systems*, 36:45716–45743, 2023a.

Sotetsu Koyamada, Paavo Parmas, Tadashi Kozuno, and Shin Ishii. Emergence of exploration in policy gradient reinforcement learning via resetting. OpenReview submission to ICLR 2023, 2023b. https://openreview.net/forum?id=GKsNIC_mQRG.

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.

Pierre L'Ecuyer. A unified view of the IPA, SF, and LR gradient estimation techniques. *Management Science*, 36(11):1364–1383, 1990.

Liu Leqi, Audrey Huang, Zachary Lipton, and Kamyar Azizzadenesheli. Supervised learning with general risk functionals. In *International Conference on Machine Learning*, pp. 12570–12592. PMLR, 2022.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, 2022.

Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. Jointly reinforcing diversity and quality in language model generations. *arXiv preprint arXiv:2509.02534*, 2025.

Zhenwen Liang, Sidi Lu, Wenhao Yu, Kishan Panaganti, Yujun Zhou, Haitao Mi, and Dong Yu. Can LLMs guide their own exploration? gradient-guided reinforcement learning for LLM reasoning. *arXiv preprint arXiv:2512.15687*, 2025.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Conference on Language Modeling (COLM)*, 2025.

Gábor Lugosi and Shahar Mendelson. Robust multivariate mean estimation: The optimality of trimmed mean. *The Annals of Statistics*, 49(1):393–410, 2021. doi: 10.1214/20-AOS1961.

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*, volume 38, 2025.

Kohsei Matsutani, Shota Takashiro, Gouki Minegishi, Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. RL squeezes, SFT expands: A comparative study of reasoning LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

Andreas Maurer, Daniela Angela Parletta, Andrea Paudice, and Massimiliano Pontil. Robust unsupervised learning via L-statistic minimization. In *International Conference on Machine Learning*, pp. 7524–7533. PMLR, 2021.

Borislav Mavrin, Shangtong Zhang, Hengshuai Yao, Linglong Kong, Kaiwen Wu, and Yaoliang Yu. Distributional reinforcement learning for efficient exploration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4424–4434. PMLR, 2019.

Andriy Mnih and Danilo J. Rezende. Variational inference for monte carlo objectives. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2188–2196, 2016.

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62, 2020.

Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. Nonparametric return distribution approximation for reinforcement learning. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 799–806, 2010a.

Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. Parametric return density estimation for reinforcement learning. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pp. 368–375, 2010b.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021. doi: 10.48550/arXiv.2112.09332.

Thanh Nguyen-Tang, Sunil Gupta, and Svetha Venkatesh. Distributional reinforcement learning via moment matching. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, volume 35, pp. 9144–9152, 2021.

Soichiro Nishimori, Paavo Parmas, Sotetsu Koyamada, Tadashi Kozuno, Toshinori Kitamura, Shin Ishii, and Yutaka Matsuo. Emergence of exploration in policy gradient reinforcement learning via retrying. In *Proceedings of the International Conference on Machine Learning*, 2026.

Wlodzimierz Ogryczak and Arie Tamir. Minimizing the sum of the k largest functions in linear time. *Information Processing Letters*, 85(3):117–122, 2003.

Ben O'Neill. The distribution of order statistics under sampling without replacement. *Journal of Statistical Theory and Applications*, 24:663–698, 2025. doi: 10.1007/s44199-025-00125-y.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. doi: 10.1177/0278364919887447.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Paavo Parmas and Takuma Seno. Proppo: A message passing framework for customizable and composable learning algorithms. *Advances in Neural Information Processing Systems*, 35:29152–29165, 2022.

Paavo Parmas and Masashi Sugiyama. A unified view of likelihood ratio and reparameterization gradients. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 4078–4086, 2021.

Paavo Parmas, Carl Edward Rasmussen, Jan Peters, and Kenji Doya. PIPPS: Flexible model-based policy search robust to the curse of chaos. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4065–4074, 2018.

Ruotian Peng, Yi Ren, Zhouliang Yu, Weiyang Liu, and Yandong Wen. SimKO: Simple Pass@K policy optimization. *arXiv preprint arXiv:2510.14807*, 2025.

Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008. doi: 10.1016/j.neunet.2008.02.003.

Shrinivas Ramasubramanian, Daman Arora, Fahim Tajwar, Guanning Zeng, Qingyang Wu, Zhongzhu Zhou, Chenfeng Xu, Haiwen Feng, Yuda Song, Aarti Singh, Ruslan Salakhutdinov, J. Andrew Bagnell, Jeff Schneider, and Andrea Zanette. Tail-likelihood reinforcement learning. *arXiv preprint arXiv:2609.02987*, 2026. URL https://arxiv.org/abs/2609.02987.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1278–1286, 2014.

R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000.

R. Tyrrell Rockafellar and Stanislav Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 26(7):1443–1471, 2002.

Mark Rowland, Marc G. Bellemare, Will Dabney, Rémi Munos, and Yee Whye Teh. An analysis of categorical distributional reinforcement learning. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 29–37. PMLR, 2018.

Mark Rowland, Robert Dadashi, Saurabh Kumar, Rémi Munos, Marc G. Bellemare, and Will Dabney. Statistics and samples in distributional reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5528–5536. PMLR, 2019.

Amrith Setlur, Matthew YR Yang, Charlie Snell, Jeremy Greer, Ian Wu, Virginia Smith, Max Simchowitz, and Aviral Kumar. e3: Learning to explore enables extrapolation of test-time compute for llms. *arXiv preprint arXiv:2506.09026*, 2025.

Darsh J Shah, Peter Rushton, Somanshu Singla, Mohit Parmar, Kurt Smith, Yash Vanjani, Ashish Vaswani, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Anthony Polloreno, Ashish Tanwer, Burhan Drak Sibai, Divya S Mansingka, Divya Shivaprasad, Ishaan Shah, Karl Stratos, Khoi Nguyen, Michael Callahan, Michael Pust, Mrinal Iyer, Philip Monk, Platon Mazarakis, Ritvik Kapila, Saurabh Srivastava, and Tim Romanski. Rethinking reflection in pre-Training. *arXiv preprint arXiv:2504.04022*, 2025.

Shai Shalev-Shwartz and Yonatan Wexler. Minimizing the maximal loss: How and why. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 793–801, 2016.

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

Alexander Shapiro. On Kusuoka representation of law invariant risk measures. *Mathematics of Operations Research*, 38(1):142–152, 2013.

Han Shen. On entropy control in LLM-RL algorithms. In *The Fourteenth International Conference on Learning Representations*, 2026.

Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory W. Wornell, Subhro Das, David Daniel Cox, and Chuang Gan. Satori: Reinforcement learning teaches large language models to reason with search. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54663–54700. PMLR, 2025.

Yanyao Shen and Sujay Sanghavi. Learning with bad training data via iterative trimmed loss minimization. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5739–5748. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/shen19e.html.

Yuda Song, Julia Kempe, and Remi Munos. Outcome-based exploration for LLM reasoning. *arXiv preprint arXiv:2509.06941*, 2025.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M. 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*, volume 33, pp. 3008–3021. Curran Associates, Inc., 2020.

Masashi Sugiyama, Hirotaka Hachiya, Hisashi Kashima, and Tetsuro Morimura. Least absolute policy iteration—a robust approach to value function approximation. *IEICE Transactions on Information and Systems*, E93-D(9):2555–2565, 2010.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop Overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*, 2025. https://openreview.net/forum?id=HvoG8SxggZ.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.02710.

Shota Takashiro, Soichiro Nishimori, Paavo Parmas, Yongmin Kim, Kohsei Matsutani, Gouki Minegishi, Yusuke Iwasawa, Takeshi Kojima, and Yutaka Matsuo. On advantage estimates for Max@K policy gradients. *arXiv preprint arXiv:2606.06080*, 2026. URL https://arxiv.org/abs/2606.06080.

Aviv Tamar, Yonatan Glassner, and Shie Mannor. Optimizing the CVaR via sampling. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. doi: 10.1609/aaai.v29i1.9561. URL https://ojs.aaai.org/index.php/AAAI/article/view/9561.

Yunhao Tang, Kunhao Zheng, Gabriel Synnaeve, and Remi Munos. Optimizing language models for inference time objectives using reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 59066–59085. PMLR, 2025.

Jens Tuyls, Dylan J Foster, Akshay Krishnamurthy, and Jordan T Ash. Representation-based exploration for language models: From test-time to post-training. *arXiv preprint arXiv:2510.11686*, 2025.

Christian Walder and Deep Tejas Karkhanis. Pass@k policy optimization: Solving harder reinforcement learning problems. *Advances in Neural Information Processing Systems*, 38:152416–152445, 2025.

Kaiwen Wang, Dawen Liang, Nathan Kallus, and Wen Sun. A reductions approach to risk-sensitive reinforcement learning with optimized certainty equivalents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63636–63661. PMLR, 2025a.

Xinran Wang, Jin Du, Azal Ahmad Khan, Qi Le, Enmao Diao, Jiawei Zhou, Jie Ding, and Ali Anwar. Beyond expectations: Quantile-guided alignment for risk-calibrated language models. In *Advances in Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=R7HJj1YvJH.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. OctoThinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025c.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jGbRWwIidy.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why RLVR may or may not escape its origin. *arXiv preprint arXiv:2507.14843*, 2025.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Derek Yang, Li Zhao, Zichuan Lin, Tao Qin, Jiang Bian, and Tie-Yan Liu. Fully parameterized quantile function for distributional reinforcement learning. In *Advances in Neural Information Processing Systems 32*, 2019.

Kenny Young and Tian Tian. MinAtar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2025.

Zhaoning Yu, Zhaolun Su, Leitian Tao, Haozhu Wang, Aashu Singh, Hanchao Yu, Jianyu Wang, Hongyang Gao, Weizhe Yuan, Jason E. Weston, Ping Yu, and Jing Xu. RESTRAIN: From spurious votes to signals — self-training RL with self-penalization. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Charlie Zhang, Graham Neubig, and Xiang Yue. On the interplay of pre-training, mid-training, and rl on reasoning language models. *arXiv preprint arXiv:2512.07783*, 2025a.

Shenao Zhang, Donghan Yu, Yihao Feng, Bowen Jin, Zhaoran Wang, John Peebles, and Zirui Wang. Learning to reason as action abstractions with scalable mid-training rl. *arXiv preprint arXiv:2509.25810*, 2025b.

Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: Rl post-training amplifies behaviors learned in pretraining. In *Second Conference on Language Modeling*, 2025.

Tianyu Zheng, Tianshun Xing, Qingshui Gu, Taoran Liang, Xingwei Qu, Xin Zhou, Yizhi Li, Zhoufutu Wen, Chenghua Lin, Wenhao Huang, et al. First return, entropy-eliciting explore. *arXiv preprint arXiv:2507.07017*, 2025.

Jin Peng Zhou, Kaiwen Wang, Jonathan Chang, Zhaolin Gao, Nathan Kallus, Kilian Q. Weinberger, Kianté Brantley, and Wen Sun. $Q\sharp$: Provably optimal distributional RL for LLM post-training. In *Advances in Neural Information Processing Systems*, volume 38, 2025a.

Yujun Zhou, Zhenwen Liang, Haolin Liu, Wenhao Yu, Kishan Panaganti, Linfeng Song, Dian Yu, Xiangliang Zhang, Haitao Mi, and Dong Yu. Evolving language models without labels: Majority drives selection, novelty promotes variation. *arXiv preprint arXiv:2509.15194*, 2025b.

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.

Appendix

A Extended Related Work **21**

B Notation **23**

C Exact known-$(\bar{R}, p)$ formulas **24**

C.1 Unconditional order statistics 24

C.2 Conditional-on-first-draw order statistics 24

C.3 Mixture identity, advantages, and L-statistics 25

C.4 Known-regime likelihood-ratio gradient 25

D Batch values and advantage computation **26**

D.1 U-statistic values 26

D.2 Sort-plus-weights formula for values 26

D.3 Include-one and leave-one-out definitions 26

D.4 Include-one expectation 27

D.5 Leave-one-out expectation 27

D.6 Prefix/suffix implementation 28

E Advantage equivalence proof **28**

F Gradient estimators from batch quantities **29**

F.1 Likelihood-ratio estimator 29

F.2 Reparameterization estimator 29

G Computation, regularity, and ties **29**

G.1 Collapsed L-statistic computation 29

G.2 Continuous rewards and ties 30

H Additional Order-Statistic Weight Schemes **30**

H.1 Weight profiles 30

H.2 CVaR gradients and finite lower-tail objectives 35

H.3 Harmonic weights: MaxRL and TailRL 35

I Additional Toy Illustrative Experiments **36**

I.1 Illustrative Experiment on Portfolio Management using CVaR 36

I.2 Robust regression with an order-statistic training objective 40

J LLM Experiments **41**

J.1 Setting 41

J.2 Additional Results 42

K RL Experiments **45**

A Extended Related Work

Learning criteria beyond the mean. This work is motivated by the view that learning should target desirable properties of the loss or reward distribution, not only the mean (Holland & Tanabe, 2023). Order-based criteria sort outcomes and weight them by rank; these include L-statistics and L-estimates from robust statistics (Daniell, 1920; Bickel & Lehmann, 1975; Huber & Ronchetti, 2009), robust L-statistic learning (Maurer et al., 2021), and rank-weighted learning theory (Khim et al., 2020; Leqi et al., 2022).

Risk measures, quantiles, CVaR, and spectral risks. CVaR/expected shortfall and spectral risk measures can be written as weighted integrals of quantiles (Acerbi, 2002; Acerbi & Tasche, 2002b;a; Rockafellar & Uryasev, 2000; 2002; Shapiro, 2013). Risk-sensitive and CVaR optimization have been studied in stochastic optimization and RL (Chow et al., 2015; 2018; Cardoso & Xu, 2019; Curi et al., 2020; Holland & Haress, 2021; 2022). Recent work also studies broader families of law-sensitive objectives, such as optimized certainty equivalents, which include CVaR, entropic risk, and mean–variance criteria and can be handled by reductions to risk-neutral RL or policy-gradient subroutines (Wang et al., 2025a). In LLM alignment, quantile-guided and quantile-reward methods similarly target specific parts of the output-reward distribution rather than only the mean (Wang et al., 2025b; Matrenok et al., 2025). OrderGrad differs in focusing on a finite-k gradient estimator that can target arbitrary L-statistic weights directly.

Distributional reinforcement learning. Distributional reinforcement learning treats the discounted return as a random variable and models its probability law rather than collapsing it immediately to an expected value. Early work used such distributional information for robustness and risk-aware decision making before the modern deep-RL formulation of the area emerged. For instance, Sugiyama et al. (2010) introduced least absolute policy iteration as a robust counterpart to least-squares value-function approximation, while Morimura et al. (2010b) and Morimura et al. (2010a) estimated return densities and return distributions directly. This perspective is closely related to risk-sensitive RL, but the emphasis is different: risk-sensitive methods usually specify a scalar functional in advance, such as a variance penalty, percentile, CVaR, or an optimized certainty equivalent, whereas distributional RL first represents a richer object—the return law—from which many such functionals can be extracted.

The contemporary deep distributional-RL literature was largely shaped by the distributional Bellman formulation and the C51 algorithm of Bellemare et al. (2017). Subsequent analysis clarified the categorical projection and its connection to the Cramér distance (Rowland et al., 2018). A complementary family of methods represents the quantile function, or inverse CDF, rather than a categorical distribution: QR-DQN learns fixed quantile locations by quantile regression (Dabney et al., 2018b), IQN conditions on sampled quantile fractions to obtain an implicit quantile-function representation (Dabney et al., 2018a), and FQF learns the quantile fractions themselves (Yang et al., 2019). These quantile-based approaches are especially relevant to OrderGrad because L-statistic and spectral objectives can be written as weighted integrals over quantiles.

Distributional RL has since been extended in both theory and algorithms. D4PG incorporated distributional critics into off-policy actor-critic learning for continuous control (Barth-Maron et al., 2018), and Rainbow showed that a categorical distributional update combines effectively with other improvements to deep Q-learning (Hessel et al., 2018). A theoretical line investigates which summaries of the return distribution are propagated by distributional dynamic programming, including statistics-and-samples formulations and moment-matching approaches (Rowland et al., 2019; Nguyen-Tang et al., 2021). More recent work sharpens this theory: Cho et al. (2025) study distributional RL with general value-function approximation and characterize Bellman-unbiased functionals; Kastner et al. (2025) analyze categorical distributional RL with KL losses; and Jhaveri et al. (2025) prove convergence guarantees for entropy-regularized policies and their induced return distributions. At the algorithmic level, distributional estimates have been used for exploration by exploiting upper-tail behavior or variability in the learned return distribution (Mavrin et al., 2019), while Dong et al. (2026) use flow matching to estimate full return distributions beyond fixed bins or quantiles. For a comprehensive account of the mathematical and algorithmic foundations of the field, see Bellemare et al. (2023).

OrderGrad is complementary to this body of work. Distributional-RL methods typically learn a critic that represents a return distribution, or selected statistics of that distribution, and the resulting policy may still be selected by maximizing expected return. OrderGrad instead gives a direct policy-gradient estimator for a specified finite-k order-statistic objective. Thus, a distributional critic could help estimate, interpret, or choose the reward distribution being targeted, but OrderGrad does not require a parametric model of the full return law. Conversely, OrderGrad supplies a policy-gradient route for directly optimizing deployment-oriented criteria such as Top-M@K, best-of-K, lower-tail CVaR-like averages, medians, trimmed means, and other rank-weighted objectives that distributional RL is often able to represent but does not necessarily optimize directly. This complementarity is also emerging in LLM post-training: for example, $Q\sharp$ uses distributional RL to learn regularized value information for LLM post-training (Zhou et al., 2025a), whereas OrderGrad changes the rank-weighted objective optimized by the policy-gradient signal itself.

Top-k, maximum losses, and best-of-k training. Average top-k and maximum-loss objectives appear in supervised learning and robust optimization (Fan et al., 2017; Shalev-Shwartz & Wexler, 2016; Ogryczak & Tamir, 2003). In language-model and policy-gradient settings, recent work has studied objectives that explicitly match inference-time selection or pass@k behavior (Koyamada et al., 2023b; Nishimori et al., 2026; Tang et al., 2025; Walder & Karkhanis, 2025; Chen et al., 2025b; Bagirov et al., 2025; Peng et al., 2025). These methods are closest to OrderGrad when the only emphasized rank is the maximum. OrderGrad treats Max@k, Top-M@k, lower-tail, median, trimmed, and signed rank-weighted criteria as instances of the same finite-sample L-statistic gradient framework. The main distinction is therefore not the use of sorted samples itself, but the generality of the rank-weight vector and the include-one/leave-one-out advantage that makes arbitrary finite-k L-statistics usable inside standard policy-gradient or generative-model updates.

Gradient estimators. Score-function, infinitesimal perturbation, and pathwise derivative estimators are classical tools for stochastic simulation and learning (L'Ecuyer, 1990; Williams, 1992; Mohamed et al., 2020). Reparameterization gradients are standard in latent-variable generative models (Kingma & Welling, 2014; Rezende et al., 2014), and recent work studies unified or composite estimators that combine LR and RP signals (Parmas & Sugiyama, 2021; Parmas & Seno, 2022). OrderGrad is complementary: it contributes the conditional order-statistic advantage needed to apply these estimator families to L-statistic objectives.

RLVR in LLMs. Reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2025; Guo et al., 2025a; Yu et al., 2025) optimizes LLM policies using deterministic feedback from objective verifiers rather than subjective human preference (Christiano et al., 2017). Recent work gives a nuanced picture of what this training does. Some studies argue that RLVR primarily amplifies or reshapes behaviors already present in the base model (Liu et al., 2025; Zhao et al., 2025; Shah et al., 2025; Wu et al., 2025), while others find that RLVR can extend the reasoning boundary when evaluated with metrics that account for both final answers and intermediate reasoning traces (Wen et al., 2026). Studies of pass@k and reasoning diversity show that post-training can improve one-shot accuracy while reducing the diversity of high-quality solution paths (Yue et al., 2025; Dang et al., 2025; Cui et al., 2025; Matsutani et al., 2026; Hu et al., 2025; Bu et al., 2025). This has motivated work on mid-training, coverage, and search-oriented reasoning data as prerequisites or complements to effective RLVR (Shen et al., 2025; Wang et al., 2025c; Zhang et al., 2025b; Chen et al., 2025a; Zhang et al., 2025a). OrderGrad targets this same tension from the objective side: it can put gradient pressure on upper-tail discovery, average top-M behavior, or lower-tail/length control without changing the underlying RL algorithm.

Quantile, distributional, and rank-based alignment for LLMs. Several recent LLM-alignment methods also move beyond mean reward. K-order ranking preference optimization uses higher-order ranking information instead of only pairwise preferences (Cai et al., 2025). Quantile-guided alignment explicitly controls targeted reward quantiles to reduce rare catastrophic outputs while preserving average performance (Wang et al., 2025b), and quantile reward policy optimization uses quantile rewards to obtain an offline, pointwise-reward policy-optimization route with reduced length bias (Matrenok et al., 2025). Inference-time-objective training directly optimizes metrics such as pass@k and majority-vote success (Tang et al., 2025), while value-based distributional approaches such as $Q\sharp$ learn distributional value information for KL-regularized LLM post-training (Zhou et al., 2025a).

OrderGrad is complementary to these directions: it provides an unbiased finite-sample policy-gradient estimator for an explicitly chosen L-statistic, so the desired rank pressure can be imposed as an advantage transformation rather than only through reranking, scalar reward shaping, or a separately learned distributional critic.

Exploration in RL for LLMs. Because RLVR can collapse output diversity or over-concentrate probability mass on a narrow set of trajectories, prior studies have incorporated exploration into LLM RL. Entropy-based approaches encourage policy uncertainty or explicitly control entropy during training (Cui et al., 2025; Cheng et al., 2025; Zheng et al., 2025; Shen, 2026; Jiang et al., 2025). Other methods use outcome-level or trajectory-level novelty: UCB-style bonuses for outcomes (Song et al., 2025), self-penalization for high-confidence deviations from majority consensus (Yu et al., 2026), up-weighting low-probability correct trajectories (He et al., 2025), Random Network Distillation for unknown trajectories (Gao et al., 2025; Burda et al., 2018), semantic or embedding-based novelty scores (Zhou et al., 2025b; Li et al., 2025), and representation-based novelty from hidden states (Tuyls et al., 2025). Additional work uses reward-model gradients to guide sampling temperature (Liang et al., 2025) or promotes in-context exploration via skill asymmetries and negative gradients (Setlur et al., 2025). OrderGrad is orthogonal to these mechanisms: instead of adding an exploration bonus, it changes which ranks in a rollout group generate gradient pressure. Increasing K or emphasizing Top-M ranks can encourage discovery under best-of-K deployment, while combining top correctness rewards with bottom length or risk rewards can discourage pathological tails.

B NOTATION

This appendix gives detailed derivations behind the main text. We follow the organization: first an exact known-$(\bar{R}, p)$ regime, then a sampling regime with an i.i.d. batch. The order-statistic sample size is k and the optimization batch size is N.

Regime A: known finite distribution. There are m arms. Arm $b \in \{1, \dots, m\}$ has deterministic reward $\bar{R}_b \in \mathbb{R}$ and sampling probability p_b, with $p \in \Delta^{m-1}$. We draw

$$A_1, \dots, A_k \overset{i.i.d.}{\sim} p, \qquad R_\ell \coloneqq \bar{R}_{A_\ell}, \qquad R_{(1:k)} \le \cdots \le R_{(k:k)}.$$

For each rank $j \in \{1, \dots, k\}$, the exact population value, conditional value, and arm advantage are

$$\bar{v}_j \coloneqq \mathbb{E}[R_{(j:k)}], \qquad \bar{q}_{b,j} \coloneqq \mathbb{E}[R_{(j:k)} \mid A_1 = b], \qquad \bar{a}_{b,j} \coloneqq \bar{q}_{b,j} - \bar{v}_j.$$

For L-statistic weights $\alpha \in \mathbb{R}^k$,

$$\bar{v}^\alpha \coloneqq \sum_{j=1}^{k} \alpha_j \bar{v}_j, \qquad \bar{q}_b^{(\alpha)} \coloneqq \sum_{j=1}^{k} \alpha_j \bar{q}_{b,j}, \qquad \bar{a}_b^{(\alpha)} \coloneqq \sum_{j=1}^{k} \alpha_j \bar{a}_{b,j}.$$

Regime B: sampled batch. In the sampling regime we observe

$$A^{(1)}, \dots, A^{(N)} \overset{i.i.d.}{\sim} p, \qquad R_i \coloneqq \bar{R}_{A^{(i)}}.$$

Batch-computable quantities are unbarred. The value estimator v_j is defined for $1 \le k \le N$; the LR leave-one-out advantage uses $1 \le k < N$. Within this range, v_j is the U-statistic value estimator, $q_{i,j}$ is the include-one subset expectation, $v_j^{(-i)}$ is the leave-one-out subset expectation, and

$$a_{i,j} \coloneqq q_{i,j} - v_j^{(-i)}, \qquad v^\alpha \coloneqq \sum_{j=1}^{k} \alpha_j v_j, \qquad a_i^{(\alpha)} \coloneqq \sum_{j=1}^{k} \alpha_j a_{i,j}.$$

Ties. Equal rewards are not collapsed. We use a stable total ordering of arms or batch positions. With ties, intermediate CDF tables should be interpreted as stable-rank CDFs rather than numeric CDFs; because tied entries have identical rewards, the final expectations remain unchanged.

BINOMIAL-TAIL CONVENTION

Throughout, $\mathrm{Bin}(n, \pi)$ denotes a binomial random variable, and

$$\mathbb{P}(\mathrm{Bin}(n, \pi) \ge s) = 1 \quad (s \le 0), \qquad \mathbb{P}(\mathrm{Bin}(n, \pi) \ge s) = 0 \quad (s > n).$$

C Exact known-$(\bar{R}, p)$ formulas

Let π be a stable sorting permutation of arms:

$$\bar{R}_{(1)} \le \bar{R}_{(2)} \le \cdots \le \bar{R}_{(m)}, \qquad p_{(t)} \coloneqq p_{\pi(t)}.$$

Let $\rho(b)$ be the stable sorted position of arm b, so that $\pi(\rho(b)) = b$. Define the stable-rank CDF grid

$$F_0 \coloneqq 0, \qquad F_t \coloneqq \sum_{s=1}^{t} p_{(s)}, \qquad t = 1, \dots, m.$$

When all rewards are distinct, $F_t = \mathbb{P}(R \le \bar{R}_{(t)})$. With ties, F_t is the probability that the sampled arm lies among the first t entries of the stable sorted list.

C.1 Unconditional order statistics

Fix $j \in \{1, \dots, k\}$. For each stable threshold t, define the count

$$Y_t \coloneqq \#\{\ell \in \{1, \dots, k\} : \rho(A_\ell) \le t\}.$$

Then $Y_t \sim \mathrm{Bin}(k, F_t)$.

Lemma C.1 (Order-statistic CDF as a binomial tail)**.** *For each $t = 0, \dots, m$,*

$$\mathbb{P}\{\textit{the stable rank of } R_{(j:k)} \textit{ is at most } t\} = \mathbb{P}\{\mathrm{Bin}(k, F_t) \ge j\}.$$

When rewards are distinct, this event is exactly $\{R_{(j:k)} \le \bar{R}_{(t)}\}$.

Proof. The jth order statistic has stable rank at most t if and only if at least j of the k sampled arms have stable rank at most t. That count is Y_t, which is binomial with parameters (k, F_t). □

Define the tail table

$$T^{(k)}_{t,j} \coloneqq \mathbb{P}\{\mathrm{Bin}(k, F_t) \ge j\}, \qquad t = 0, \dots, m, \quad j = 1, \dots, k. \tag{24}$$

Differencing adjacent CDF values gives the probability that stable sorted entry t supplies the jth order statistic:

$$W^{\mathrm{wr}}_{t,j} \coloneqq T^{(k)}_{t,j} - T^{(k)}_{t-1,j}, \qquad t = 1, \dots, m. \tag{25}$$

Therefore

$$\bar{v}_j = \sum_{t=1}^{m} \bar{R}_{(t)} W^{\mathrm{wr}}_{t,j}. \tag{26}$$

Matrix form. Let $\bar{R}_{(\cdot)} = (\bar{R}_{(1)}, \dots, \bar{R}_{(m)})^\top$. Let $T^{(k)} \in \mathbb{R}^{(m+1)\times k}$ collect (24), and let $D \in \mathbb{R}^{m\times(m+1)}$ be the forward-difference operator

$$D_{t,t} = 1, \qquad D_{t,t-1} = -1, \qquad \text{all other entries } 0.$$

Then $W^{\mathrm{wr}} = DT^{(k)}$ and

$$\bar{v} = (\bar{v}_1, \dots, \bar{v}_k) = \bar{R}^\top_{(\cdot)} W^{\mathrm{wr}} = \bar{R}^\top_{(\cdot)} D T^{(k)}. \tag{27}$$

C.2 Conditional-on-first-draw order statistics

Fix arm b. Define the stable-rank indicator

$$\delta_{b,t} \coloneqq \mathbf{1}\{\rho(b) \le t\}.$$

Conditioning on $A_1 = b$ pins one draw and leaves $A_2, \dots, A_k$ i.i.d. For a stable threshold t, the pinned draw contributes $\delta_{b,t}$ to the count below the threshold, while the remaining $k-1$ draws contribute a binomial count with success probability F_t. Hence

$$\mathbb{P}\{\text{stable rank of } R_{(j:k)} \text{ is at most } t \mid A_1 = b\} = \mathbb{P}\{\mathrm{Bin}(k-1, F_t) \ge j - \delta_{b,t}\}. \tag{28}$$

Define

$$T_{t,j}^{(b)} \coloneqq \mathbb{P}\{\mathrm{Bin}(k-1, F_t) \geq j - \delta_{b,t}\}. \tag{29}$$

The conditional mass table is

$$W_{t,j}^{(b)} \coloneqq T_{t,j}^{(b)} - T_{t-1,j}^{(b)}, \tag{30}$$

and therefore

$$\bar{q}_{b,j} = \sum_{t=1}^{m} \bar{R}_{(t)} W_{t,j}^{(b)}. \tag{31}$$

In matrix form,

$$\bar{q}_b = (\bar{q}_{b,1}, \ldots, \bar{q}_{b,k}) = \bar{R}_{(\cdot)}^{\top} W^{(b)} = \bar{R}_{(\cdot)}^{\top} D T^{(b)}. \tag{32}$$

Shared precomputation. Let

$$G_{t,s} \coloneqq \mathbb{P}\{\mathrm{Bin}(k-1, F_t) \geq s\}.$$

Then G can be precomputed once for all t and integer thresholds s, and

$$T_{t,j}^{(b)} = G_{t,j-\delta_{b,t}}.$$

Conditioning on arm b only shifts the rank threshold by 0 or 1 at each stable sorted grid point.

C.3 Mixture identity, advantages, and L-statistics

Proposition C.2 (Mixture identity)**.** *For every rank j,*

$$\bar{v}_j = \sum_{b=1}^{m} p_b \bar{q}_{b,j}.$$

Proof. By the law of total expectation over the first sampled arm,

$$\mathbb{E}[R_{(j:k)}] = \sum_{b=1}^{m} \mathbb{E}[R_{(j:k)} \mid A_1 = b] \mathbb{P}(A_1 = b) = \sum_{b=1}^{m} p_b \bar{q}_{b,j}.$$

□

The exact arm advantage is

$$\bar{a}_{b,j} \coloneqq \bar{q}_{b,j} - \bar{v}_j, \qquad \bar{a}_b = (\bar{a}_{b,1}, \ldots, \bar{a}_{b,k}). \tag{33}$$

Consequently $\sum_{b=1}^{m} p_b \bar{a}_{b,j} = 0$ for every j. For an L-statistic,

$$\bar{v}^{\alpha} = \alpha^{\top} \bar{v}, \qquad \bar{q}_b^{(\alpha)} = \alpha^{\top} \bar{q}_b, \qquad \bar{a}_b^{(\alpha)} = \alpha^{\top} \bar{a}_b. \tag{34}$$

C.4 Known-regime likelihood-ratio gradient

Assume the rewards $\bar{R}_b$ are fixed and the probabilities $p_\theta(b)$ are differentiable with common support. Let

$$T_\alpha(A_{1:k}) \coloneqq \sum_{j=1}^{k} \alpha_j R_{(j:k)}.$$

Then $\bar{v}^{\alpha}(\theta) = \mathbb{E}_\theta[T_\alpha(A_{1:k})]$. The score-function identity gives

$$\nabla_\theta \bar{v}^{\alpha}(\theta) = \mathbb{E}_\theta \left[T_\alpha(A_{1:k}) \sum_{\ell=1}^{k} \nabla_\theta \log p_\theta(A_\ell) \right] \tag{35}$$

$$= k\, \mathbb{E}_\theta \left[T_\alpha(A_{1:k}) \nabla_\theta \log p_\theta(A_1) \right] \tag{36}$$

$$= k \sum_{b=1}^{m} p_\theta(b) \bar{q}_b^{(\alpha)} \nabla_\theta \log p_\theta(b). \tag{37}$$

Because $\sum_b p_\theta(b) \nabla_\theta \log p_\theta(b) = 0$, subtracting the baseline $\bar{v}^{\alpha}$ gives

$$\nabla_\theta \bar{v}^{\alpha}(\theta) = k \sum_{b=1}^{m} p_\theta(b) \bar{a}_b^{(\alpha)} \nabla_\theta \log p_\theta(b). \tag{38}$$

D BATCH VALUES AND ADVANTAGE COMPUTATION

Throughout this section, the realized batch rewards $R_1, \dots, R_N$ are fixed and subset expectations are with respect to uniform subset selection without replacement. Write $[N] \coloneqq \{1, \dots, N\}$.

D.1 U-STATISTIC VALUES

For $1 \le k \le N$, define

$$v_j \coloneqq \frac{1}{\binom{N}{k}} \sum_{\substack{S \subseteq [N] \\ |S|=k}} (R_S)_{(j:k)}. \tag{39}$$

Equivalently, if S is a uniform size-k subset of $[N]$ independent of the rewards, then

$$v_j = \mathbb{E}[(R_S)_{(j:k)} \mid R_{1:N}].$$

Lemma D.1 (Random subsample of i.i.d. observations)**.** *Let $Z_1, \dots, Z_M$ be i.i.d. and let J be a uniformly random subset of $\{1, \dots, M\}$ of size s, chosen independently of $Z_{1:M}$. For any symmetric function h,*

$$\mathbb{E}[h(Z_J)] = \mathbb{E}[h(Z_1, \dots, Z_s)].$$

Proof. Condition on the subset J. Since $Z_{1:M}$ are i.i.d., the selected values have the same joint distribution as the first s values, up to ordering. Averaging over J preserves this distributional equality for symmetric functions. □

Theorem D.2 (Unbiasedness of the U-statistic value estimator)**.** *If $R_1, \dots, R_N$ are i.i.d. with the same distribution as R, then*

$$\mathbb{E}[v_j] = \bar{v}_j.$$

Proof. Let S be a uniform size-k subset independent of the rewards. Then

$$\mathbb{E}[v_j] = \mathbb{E}[(R_S)_{(j:k)}].$$

By Lemma D.1, the multiset $\{R_i : i \in S\}$ has the same distribution as k fresh i.i.d. rewards. Therefore $(R_S)_{(j:k)}$ has the same distribution as $R_{(j:k)}$, and the expectation is $\bar{v}_j$. □

D.2 SORT-PLUS-WEIGHTS FORMULA FOR VALUES

After sorting the batch as $R_{(1:N)} \le \cdots \le R_{(N:N)}$, the probability that position m supplies the jth order statistic of a uniform size-k subset is

$$W_{m,j} = \frac{\binom{m-1}{j-1}\binom{N-m}{k-j}}{\binom{N}{k}}. \tag{40}$$

Indeed, the subset must include position m, choose $j-1$ positions from the $m-1$ lower positions, and choose $k-j$ positions from the $N-m$ higher positions. Consequently

$$v_j = \sum_{m=1}^{N} R_{(m:N)} W_{m,j}, \qquad v = (v_1, \dots, v_k) = \mathbf{R}_{(\cdot)}^\top W, \tag{41}$$

where $\mathbf{R}_{(\cdot)} = (R_{(1:N)}, \dots, R_{(N:N)})^\top$. For L-statistic weights,

$$v^\alpha = \alpha^\top v = \mathbf{R}_{(\cdot)}^\top W \alpha. \tag{42}$$

D.3 INCLUDE-ONE AND LEAVE-ONE-OUT DEFINITIONS

Assume $1 \le k < N$. Let S be a uniform size-k subset of $[N]$. For each original index i,

$$q_{i,j} \coloneqq \mathbb{E}[(R_S)_{(j:k)} \mid i \in S,\ R_{1:N}], \tag{43}$$

$$v_j^{(-i)} \coloneqq \mathbb{E}[(R_{S'})_{(j:k)} \mid R_{1:N} \text{ with index } i \text{ removed}], \tag{44}$$

where S' is a uniform size-k subset of $[N] \setminus \{i\}$. Then

$$a_{i,j} \coloneqq q_{i,j} - v_j^{(-i)}. \tag{45}$$

The formulas below are written in sorted-rank space and then mapped back to original indices.

D.4 INCLUDE-ONE EXPECTATION

Let t be the sorted rank of the datapoint forced into the subset. Conditioning on including rank t means $R_{(t:N)}$ is forced into the size-k subset and the remaining $k-1$ indices are chosen uniformly from the other $N-1$ ranks.

Lemma D.3 (Include-one weights)**.** *For fixed rank j and included sorted rank t,*

$$\mathbb{P}((R_S)_{(j:k)} = R_{(m:N)} \mid t \in S, R_{1:N}) = \begin{cases} \frac{\binom{m-1}{j-1}\binom{N-m-1}{k-j-1}}{\binom{N-1}{k-1}}, & m < t, \\ \frac{\binom{t-1}{j-1}\binom{N-t}{k-j}}{\binom{N-1}{k-1}}, & m = t, \\ \frac{\binom{m-2}{j-2}\binom{N-m}{k-j}}{\binom{N-1}{k-1}}, & m > t. \end{cases}$$

Proof. We count size-k subsets containing rank t for which rank m is the jth order statistic. If $m < t$, then m must be included, $j-1$ selected ranks come from the $m-1$ lower positions, and the remaining $k-j-1$ ranks come from positions above m excluding the forced rank t. If $m = t$, the forced rank is the jth order statistic, so choose $j-1$ lower and $k-j$ higher ranks. If $m > t$, the forced rank is one of the $j-1$ lower ranks, so choose $j-2$ additional lower ranks and $k-j$ higher ranks. Divide each count by the number $\binom{N-1}{k-1}$ of size-k subsets containing t. □

Define the three precomputable matrices

$$A_{m,j} \coloneqq \frac{\binom{m-1}{j-1}\binom{N-m-1}{k-j-1}}{\binom{N-1}{k-1}}, \tag{46}$$

$$B_{m,j} \coloneqq \frac{\binom{m-1}{j-1}\binom{N-m}{k-j}}{\binom{N-1}{k-1}}, \tag{47}$$

$$C_{m,j} \coloneqq \frac{\binom{m-2}{j-2}\binom{N-m}{k-j}}{\binom{N-1}{k-1}}. \tag{48}$$

Then the include-one expectation for included sorted rank t is

$$q_{t,j} = \sum_{m<t} R_{(m:N)} A_{m,j} + R_{(t:N)} B_{t,j} + \sum_{m>t} R_{(m:N)} C_{m,j}. \tag{49}$$

D.5 LEAVE-ONE-OUT EXPECTATION

Remove sorted rank t and sample a uniform size-k subset from the remaining $N-1$ ranks. Let

$$W^-_{\ell,j} \coloneqq \frac{\binom{\ell-1}{j-1}\binom{(N-1)-\ell}{k-j}}{\binom{N-1}{k}}, \qquad \ell = 1, \dots, N-1. \tag{50}$$

After removing rank t, the reduced sorted list satisfies

$$R^{(-t)}_{(\ell:N-1)} = \begin{cases} R_{(\ell:N)}, & \ell < t, \\ R_{(\ell+1:N)}, & \ell \geq t. \end{cases}$$

Therefore

$$v_j^{(-t)} = \sum_{m<t} R_{(m:N)} W^-_{m,j} + \sum_{m>t} R_{(m:N)} W^-_{m-1,j}. \tag{51}$$

D.6 Prefix/suffix implementation

Equations (49) and (51) can be evaluated for all t and j in $O(Nk)$ time after sorting.

For include-one computation, let $M_R \coloneqq \mathbf{R}_{(\cdot)}\mathbf{1}_k^\top \in \mathbb{R}^{N\times k}$ and define

$$P_A \coloneqq M_R \odot A, \qquad P_C \coloneqq M_R \odot C,$$

where $\odot$ is the Hadamard product. With cumulative sums down rows and row 0 equal to zero,

$$\sum_{m<t} R_{(m:N)}A_{m,:} = \mathrm{cumsum}(P_A)_{t-1,:}, \qquad \sum_{m>t} R_{(m:N)}C_{m,:} = \mathrm{cumsum}(P_C)_{N,:} - \mathrm{cumsum}(P_C)_{t,:}.$$

Thus all $q_{t,:}$ are obtained by two cumulative sums and rowwise additions.

For leave-one-out computation, let $u^- = (R_{(1:N)}, \dots, R_{(N-1:N)})^\top$ and $u^+ = (R_{(2:N)}, \dots, R_{(N:N)})^\top$. Form

$$P_1 \coloneqq u^-\mathbf{1}_k^\top \odot W^-, \qquad P_2 \coloneqq u^+\mathbf{1}_k^\top \odot W^-.$$

Then

$$\begin{aligned} \sum_{m<t} R_{(m:N)}W^-_{m,:} &= \mathrm{cumsum}(P_1)_{t-1,:}, \\ \sum_{m>t} R_{(m:N)}W^-_{m-1,:} &= \mathrm{cumsum}(P_2)_{N-1,:} - \mathrm{cumsum}(P_2)_{t-1,:}. \end{aligned} \tag{52}$$

This gives $v^{(-t)}$ for all ranks t in $O(Nk)$ time. Finally,

$$a_{t,j} = q_{t,j} - v_j^{(-t)}, \qquad a_t^{(\alpha)} = \sum_{j=1}^{k} \alpha_j a_{t,j}, \tag{53}$$

and the inverse sorting permutation maps $a_{t,j}$ back to $a_{i,j}$.

E Advantage equivalence proof

Let I be uniformly random on $[N]$, independent of the dataset. Fix an arm b. The exact known-$(\bar{R}, p)$ advantage is $\bar{a}_{b,j} = \bar{q}_{b,j} - \bar{v}_j$.

Theorem E.1 (Batch advantage equivalence)**.** *Assume $1 \le k < N$. For every rank $j \in \{1, \dots, k\}$,*

$$\mathbb{E}[a_{I,j} \mid A^{(I)} = b] = \bar{a}_{b,j}.$$

Consequently, for any fixed $\alpha \in \mathbb{R}^k$,

$$\mathbb{E}[a_I^{(\alpha)} \mid A^{(I)} = b] = \bar{a}_b^{(\alpha)}.$$

Proof. We prove the include-one and leave-one-out identities separately and subtract.

Include-one term. Condition on $A^{(I)} = b$. Then $R_I = \bar{R}_b$ is fixed, and the remaining $N-1$ batch rewards are still i.i.d. with the marginal law of $R = \bar{R}_A$ under $A \sim p$. Given the dataset, $q_{I,j}$ is the expected jth order statistic of a size-k subset conditioned to include index I. Equivalently, it is the expected order statistic of the multiset consisting of the fixed reward $\bar{R}_b$ and a uniformly chosen $(k-1)$-subset of the remaining $N-1$ values. By Lemma D.1, that random subset has the same unconditional distribution as $k-1$ fresh i.i.d. draws from the reward law. Therefore

$$\mathbb{E}[q_{I,j} \mid A^{(I)} = b] = \mathbb{E}[R_{(j:k)} \mid A_1 = b] = \bar{q}_{b,j}.$$

Leave-one-out term. Again condition on $A^{(I)} = b$. Removing index I leaves $N-1$ i.i.d. rewards with the law of R. The leave-one-out quantity $v_j^{(-I)}$ is the U-statistic value estimator computed on those $N-1$ samples. By Theorem D.2, applied with $N-1$ samples,

$$\mathbb{E}[v_j^{(-I)} \mid A^{(I)} = b] = \bar{v}_j.$$

Subtract. Since $a_{I,j} = q_{I,j} - v_j^{(-I)}$,

$$\mathbb{E}[a_{I,j} \mid A^{(I)} = b] = \bar{q}_{b,j} - \bar{v}_j = \bar{a}_{b,j}.$$

Linearity gives the L-statistic statement. □

F Gradient estimators from batch quantities

F.1 Likelihood-ratio estimator

For $1 \le k < N$, the sampling-regime likelihood-ratio estimator is

$$\hat{\boldsymbol{g}}_{\text{LR-OG},N,\alpha} = \frac{k}{N} \sum_{i=1}^{N} a_i^{(\alpha)} \nabla_\theta \log p_\theta(x_i). \tag{54}$$

In the finite-action known-reward setting,

$$\begin{aligned}
\mathbb{E}[\hat{\boldsymbol{g}}_{\text{LR-OG},N,\alpha}] &= k\, \mathbb{E}\Big[a_I^{(\alpha)} \nabla_\theta \log p_\theta(A^{(I)})\Big] && (55)\\
&= k \sum_{b=1}^{m} p_\theta(b) \mathbb{E}[a_I^{(\alpha)} \mid A^{(I)} = b] \nabla_\theta \log p_\theta(b) && (56)\\
&= k \sum_{b=1}^{m} p_\theta(b) \bar{a}_b^{(\alpha)} \nabla_\theta \log p_\theta(b) && (57)\\
&= \nabla_\theta \bar{v}^\alpha(\theta), && (58)
\end{aligned}$$

where the last equality is (38). Thus the LR estimator uses the regular batch quantity $a_i^{(\alpha)}$; barred quantities appear only in the proof and in the exact known-regime formula.

F.2 Reparameterization estimator

Suppose $x_i = T_\theta(\varepsilon_i)$ with ε_i independent of θ and $R_i(\theta) = R(T_\theta(\varepsilon_i))$ differentiable. The reparameterization estimator differentiates the batch value

$$\hat{\boldsymbol{g}}_{\text{RP-OG},N,\alpha} = \nabla_\theta v^\alpha(\theta), \qquad v^\alpha(\theta) = \mathbf{R}_{(\cdot)}(\theta)^\top W \alpha. \tag{59}$$

For almost surely distinct rewards, the sorting permutation is locally constant, so

$$\hat{\boldsymbol{g}}_{\text{RP-OG},N,\alpha} = \sum_{m=1}^{N} (W\alpha)_m \nabla_\theta R_{(m:N)}(\theta). \tag{60}$$

With ties, one may use a stable-sort subgradient or a differentiable sorting relaxation. Under dominated convergence,

$$\mathbb{E}[\hat{\boldsymbol{g}}_{\text{RP-OG},N,\alpha}] = \nabla_\theta \mathbb{E}[v^\alpha(\theta)] = \nabla_\theta \bar{v}^\alpha(\theta).$$

G Computation, regularity, and ties

G.1 Collapsed L-statistic computation

The prefix/suffix formulas in App. D.6 compute every rank-specific quantity $q_{t,j}$, $v_j^{(-t)}$, and $a_{t,j}$ in $O(Nk)$ time after sorting. Often the algorithm only needs the L-statistic combination $a_t^{(\alpha)} = \sum_j \alpha_j a_{t,j}$. In that case the rank dimension can be collapsed before applying the weights to a batch.

Define the collapsed vectors

$$A_m^{(\alpha)} \coloneqq \sum_{j=1}^{k} \alpha_j A_{m,j}, \quad B_m^{(\alpha)} \coloneqq \sum_{j=1}^{k} \alpha_j B_{m,j}, \quad C_m^{(\alpha)} \coloneqq \sum_{j=1}^{k} \alpha_j C_{m,j}, \quad (W^-)_\ell^{(\alpha)} \coloneqq \sum_{j=1}^{k} \alpha_j W^-_{\ell,j}.$$

Then, in sorted-rank space,

$$q_t^{(\alpha)} = \sum_{m<t} R_{(m:N)} A_m^{(\alpha)} + R_{(t:N)} B_t^{(\alpha)} + \sum_{m>t} R_{(m:N)} C_m^{(\alpha)},$$

and

$$(v^{(-t)})^{(\alpha)} = \sum_{m<t} R_{(m:N)} (W^-)_m^{(\alpha)} + \sum_{m>t} R_{(m:N)} (W^-)_{m-1}^{(\alpha)}.$$

Thus all $a_t^{(\alpha)} = q_t^{(\alpha)} - (v^{(-t)})^{(\alpha)}$ can be computed with a constant number of prefix/suffix scans, i.e., in $O(N)$ time after sorting. For fixed N, k, and α, the collapsed weights can be precomputed once, so the per-minibatch cost is dominated by sorting and is $O(N \log N)$. If one needs all rankwise advantages or changes α online, the relevant $O(Nk)$ precomputation or rankwise computation should be counted.

The same collapse applies to the reparameterization value:

$$v^\alpha = \mathbf{R}_{(\cdot)}^\top W \alpha.$$

For fixed $W\alpha$, evaluating v^α and its pathwise gradient after sorting is linear in N.

G.2 Continuous rewards and ties

The finite-action deterministic-reward setting is useful because it gives exact barred quantities and a clean proof target for the LR estimator. The sampling-regime definitions of v_j, $q_{i,j}$, $v_j^{(-i)}$, and $a_{i,j}$ are nevertheless purely batch-level definitions and remain well-defined for continuous rewards, trajectories, completions, or latent draws.

For continuous reward distributions, ties occur with probability zero under mild non-atomicity assumptions, and the sorted-reward formulas apply directly. With atoms or equal numerical rewards, we use a stable total ordering of batch positions and do not collapse equal values. The counting formulas in App. D.2–App. D.5 count sorted positions rather than distinct reward values, so the resulting expectations remain unchanged when tied positions have equal rewards.

For the likelihood-ratio estimator, the appendix proof states the finite-action version:

$$\mathbb{E}[\hat{\boldsymbol{g}}_{\text{LR-OG},N,\alpha}] = \nabla_\theta \bar{v}^\alpha(\theta).$$

A more general trajectory-level statement follows the same score-function argument when the support is common in θ, differentiation can be interchanged with integration, and regular conditional versions of the include-one values exist. For the reparameterization estimator, the condition used in App. F.2 is the standard pathwise one: $R(T_\theta(\varepsilon))$ is differentiable and dominated convergence permits exchanging ∇_θ and $\mathbb{E}$.

H Additional Order-Statistic Weight Schemes

H.1 Weight profiles

Here we visualize additional order-statistics weights used for computing v^α. In this appendix only, plot labels of the form `Rank r` count ranks from the top: Rank 1 denotes the maximum, whereas the algebraic order-statistic index j in the main text counts from the bottom, so $j = 1$ denotes the minimum. Weights near the right side of a plot emphasize larger values, while weights near the left side emphasize smaller values. Some choices of α, such as the Gini mean difference, use signed weights to compare the two tails rather than to average a subset of order statistics.

Table 1: Summary of the order-statistic weight schemes discussed in this appendix.

Scheme	Plot label / parameterization	Meaning
Rank-r	`Rank r`	Emphasizes the rth largest value among k sampled values. Rank 1 is the maximum; rank k is the minimum.
ReMax	`ReMax` with parameter k	The maximum-of-k objective, i.e., the Rank-1 scheme. Increasing k moves mass toward the largest order statistics; $k = 1$ recovers the ordinary mean.
MaxRL / TailRL	$\alpha_j = 1/(k - j + 1)$ (unnormalized)	Finite harmonic objectives: MaxRL for binary rewards and TailRL for $R \in [0, 1]$. With $k = N - 1$, recovers their centered on-policy estimators (Section H.3).
Quantile	`Quantile:q`	Concentrates mass around the empirical q-quantile, with smaller q focusing on lower sorted indices and $q = 0.5$ corresponding to the median.
Top-M@k	`TopM:M`	Averages the top M ranks out of a size-k comparison batch.
Bottom-M@k	`BotM:M`	Averages the bottom M ranks out of a size-k comparison batch.
TopBot	`TopBot:M`	Places mass on both tails by averaging the corresponding Top-M@k and Bottom-M@k schemes.
Median	`Median`	Focuses on the central rank(s), providing a robust alternative to the mean.
Trimmed mean	`TrimM:M`	Drops the bottom M and top M ranks and averages the remaining middle ranks.
Winsorized mean	`WinsorizedM:M`	Clips the bottom and top M ranks to the nearest retained ranks before averaging, reducing tail sensitivity without discarding the tails entirely.
Gini mean difference	`GiniMeanDifference`	Uses signed weights to contrast high and low order statistics, emphasizing distributional spread rather than a location statistic.

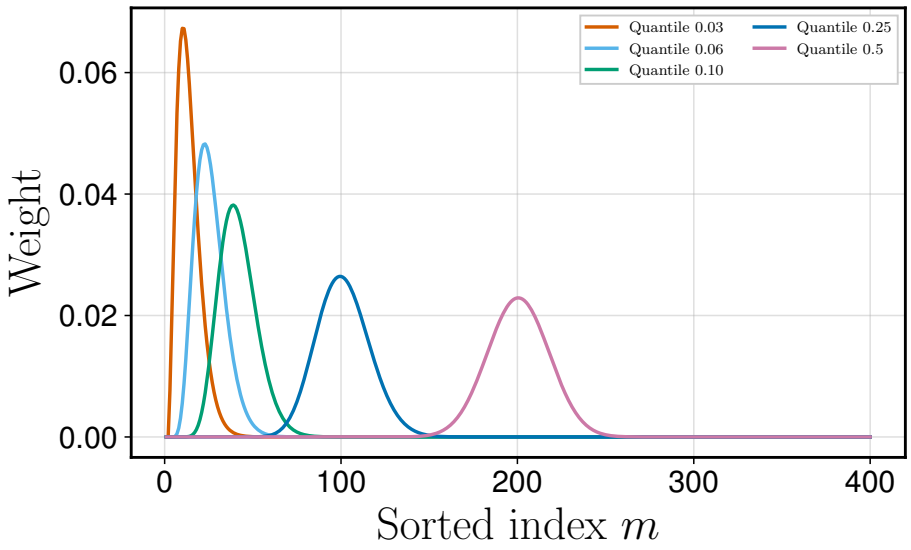


Figure 7: Quantile weight profiles for $q \in \{0.03, 0.06, 0.10, 0.25, 0.5\}$ with $N = 400$ and comparison size $k = 100$. Smaller quantiles place mass on the lower tail of the sorted batch, while the median profile is centered near $m = N/2$.

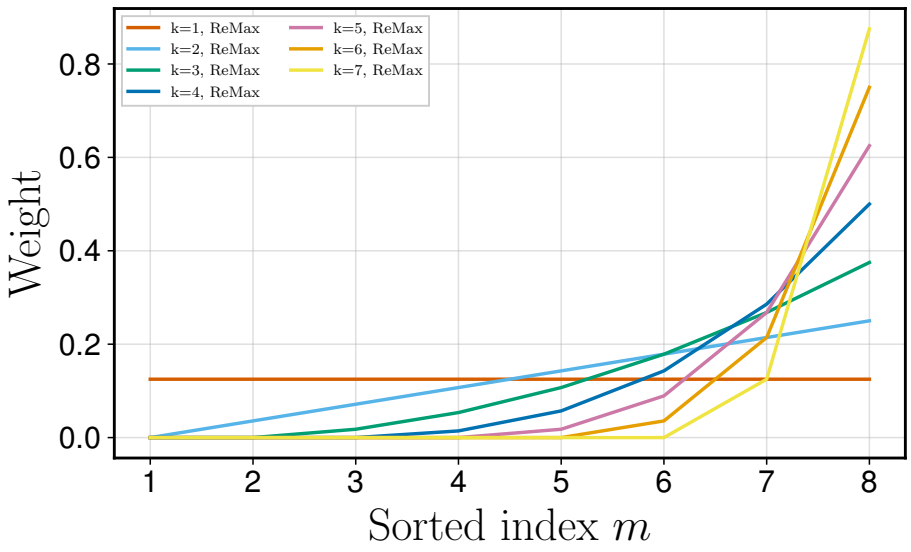


Figure 8: ReMax, or maximum-of-k, weight profiles for $N = 8$ and $k \in \{1, \ldots, 7\}$. The $k = 1$ curve is uniform, corresponding to the ordinary mean, while larger k increasingly concentrates weight on the largest sorted values.

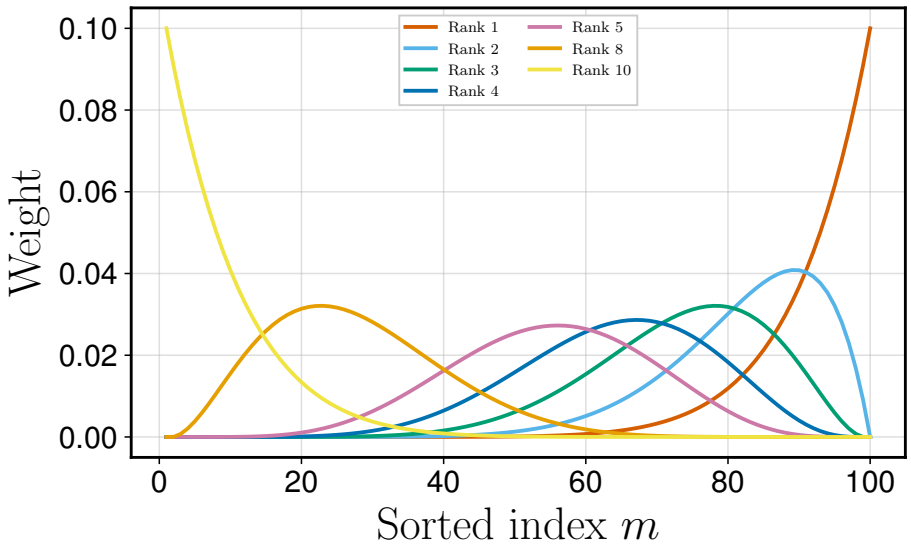


Figure 9: Rank-weight profiles for a comparison batch of size $k = 10$ with $N = 100$. Rank 1 targets the maximum of the comparison batch; increasing the rank moves the mass toward lower order statistics, with Rank 10 targeting the minimum.

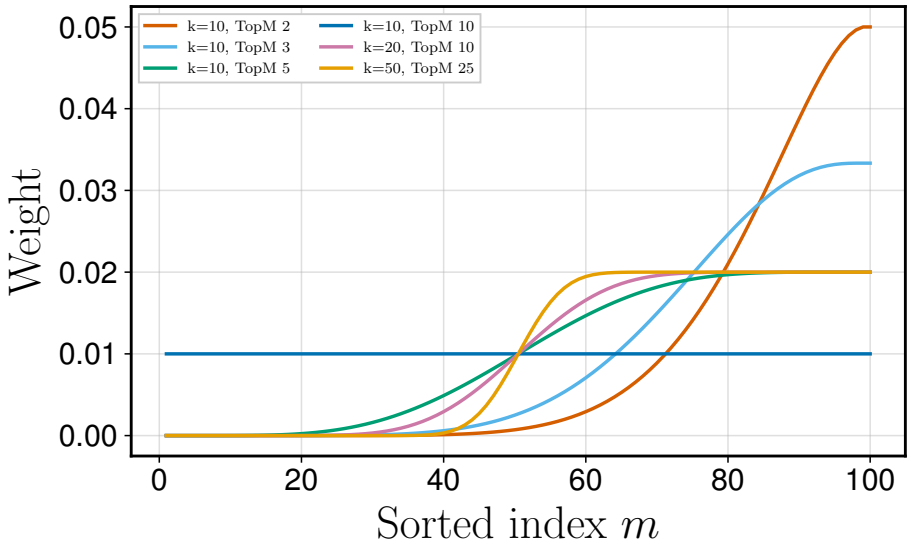


Figure 10: Top-M@k profiles for several choices of M and k with $N = 100$. These schemes interpolate between a strongly top-focused objective and the ordinary mean: when $M = k$, all ranks are averaged and the resulting profile is uniform.

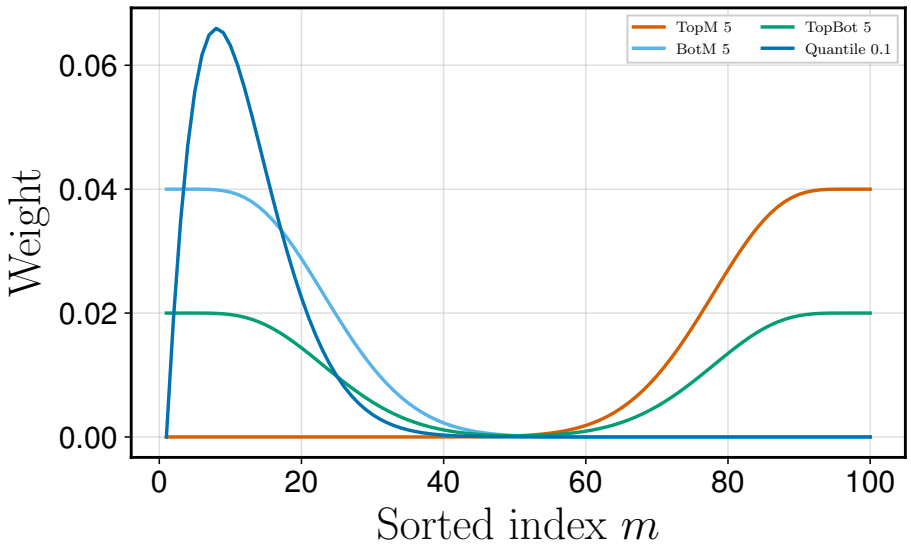


Figure 11: Tail- and quantile-focused schemes for $N = 100$ and $k = 20$. Top-M@k emphasizes high values, Bottom-M@k emphasizes low values, `TopBot` places mass on both tails, and the quantile scheme concentrates around the specified lower quantile.

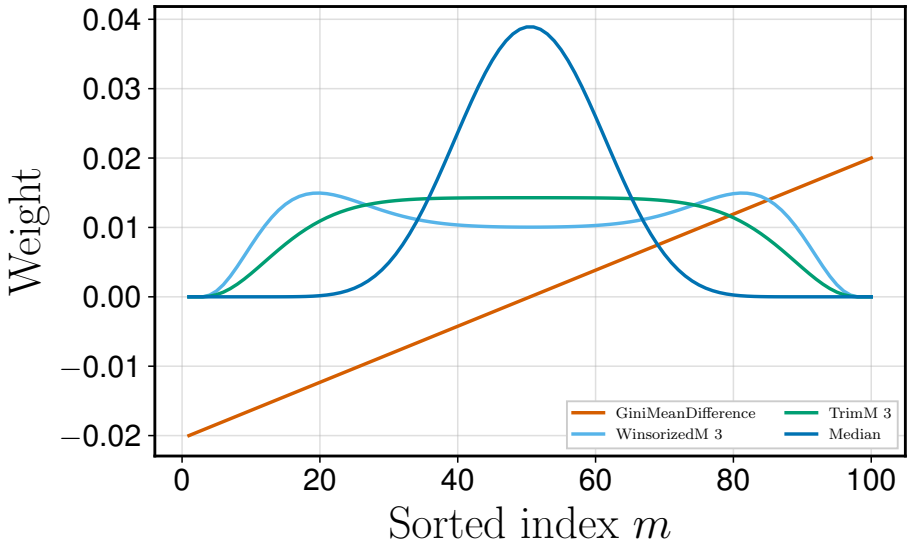


Figure 12: Robust and signed schemes for $N = 100$ and $k = 20$. The median focuses on the center, the trimmed and winsorized means reduce sensitivity to extremes, and the Gini mean difference uses signed weights to contrast the upper and lower tails.

H.2 CVaR GRADIENTS AND FINITE LOWER-TAIL OBJECTIVES

Write $S_i = \nabla_\theta \log p_\theta(x_i)$ and assume i.i.d. sampling, parameter-independent rewards, and the usual interchange of differentiation and expectation. For lower-tail probability $\rho \in (0, 1)$, let $q_\rho = Q_\theta(\rho)$ and $C_\rho = \rho^{-1} \int_0^\rho Q_\theta(u)\, du$. At a differentiable, atom-free quantile, the population-CVaR identity of Tamar et al. (2015) is

$$\nabla_\theta C_\rho = \frac{1}{\rho}\mathbb{E}\big[\min(R - q_\rho, 0)S\big]\,. \tag{61}$$

The threshold subtraction accounts for the moving quantile boundary; it cannot be replaced by an arbitrary baseline inside the selected tail.

For $1 \le M < k$, define the Bottom-M@k statistic $T_{k,M} = M^{-1}\sum_{j=1}^{M} R_{(j:k)}$ and let D_{-i} be the bottom-M average of the other $k-1$ rewards. Inserting or deleting one observation gives the exact identity

$$\begin{aligned} T_{k,M} - D_{-i} &= \frac{\min(R_i - R_{(M+1:k)}, 0)}{M}, \\ \nabla_\theta \mathbb{E}[T_{k,M}] &= \mathbb{E}\left[\frac{1}{M}\sum_{i=1}^{k} \min(R_i - R_{(M+1:k)}, 0)S_i\right]. \end{aligned} \tag{62}$$

The baseline D_{-i} excludes the scored observation, so the second identity follows from the joint score-function estimator. Thus a CVaR-like threshold update already estimates a finite lower-tail gradient; the finite target approaches C_ρ as $k \to \infty$ with $M/k \to \rho$, under suitable integrability and gradient-convergence conditions.

For $N > k$, averaging these include-one comparisons gives a signal $q_i^{(\alpha)} - D_i$, where $\alpha_j = \mathbf{1}\{j \le M\}/M$, $q_i^{(\alpha)} = \sum_j \alpha_j q_{i,j}$, $v^{\alpha,(-i)} = \sum_j \alpha_j v_j^{(-i)}$, and D_i averages the deletion baseline over comparator subsets. Following MaxPO's recomputed leave-two-out construction (Takashiro et al., 2026, Proposition 4.1 and Appendix F.3), remove i from every comparator pool before averaging the remaining signals. Subset counting makes this correction $v^{\alpha,(-i)} - D_i$, leaving exactly $a_i^{(\alpha)} = q_i^{(\alpha)} - v^{\alpha,(-i)}$. This is the connection between the threshold signal and the centered OrderGrad advantage. Averaging comparisons is Rao–Blackwellization relative to a randomized estimator with the *same* baseline, and removes conditional comparison-sampling variance. Changing the baseline need not reduce variance, and this argument does not order OrderGrad against an estimator using the exact population quantile.

The empirical-quantile convention matters. For the inverse empirical CDF, put $M = \lceil \rho N \rceil - 1 \ge 1$, $\widehat{q}_\rho = R_{(M+1:N)}$, $h_i = \min(R_i - \widehat{q}_\rho, 0)/\rho$, and $\bar{h} = N^{-1}\sum_i h_i$. The deletion identity and subset counting give

$$\begin{aligned} \mathbb{E}\left[\frac{1}{N}\sum_i h_i S_i\right] &= \frac{M}{\rho N}\nabla_\theta J_{N,\text{Bottom-}M}, \\ \frac{1}{N}\sum_i (h_i - \bar{h})S_i &= \frac{M}{\rho N}\widehat{g}_{\text{LR-OG}}^{\text{Bottom-}M@(N-1)}, \end{aligned} \tag{63}$$

where $J_{n,\text{Bottom-}M}$ denotes the expected bottom-M average of n draws. The second equality holds batch by batch, with OrderGrad evaluated on the same N observations and comparison size $N-1$: writing $T_{N,M}$ for the bottom-M average of the full batch, the deletion-set values satisfy $v^{\alpha,(-i)} = T_{N,M} - (\rho/M)h_i$, so centering cancels the common term $T_{N,M}$. Centering therefore changes the finite target from size N to $N-1$; neither expression is generally unbiased for population CVaR at finite N. If $M = 0$, both threshold estimates vanish. Interpolated quantiles yield different boundary weights.

H.3 HARMONIC WEIGHTS: MAXRL AND TAILRL

For rewards in $[0, 1]$, TailRL optimizes the integral of log upper-tail probabilities, $\int_0^1 \log \Pr_\theta(R > t)\, dt$ (Ramasubramanian et al., 2026). Its order-k truncation is a harmonic sum of expected maxima.

Writing $H_k = \sum_{\ell=1}^{k} 1/\ell$, this finite objective has the exact L-statistic representation

$$J_{\mathrm{TailRL}}^{(k)} = \sum_{\ell=1}^{k} \frac{\mathbb{E}[R_{(\ell:\ell)}] - 1}{\ell} = \mathbb{E}\left[\sum_{j=1}^{k} \frac{R_{(j:k)}}{k-j+1}\right] - H_k. \tag{64}$$

Indeed, representing each size-ℓ maximum by averaging over ℓ-subsets of k draws gives the coefficient identity $\sum_{\ell=1}^{j} \binom{j-1}{\ell-1}/[\ell\binom{k}{\ell}] = 1/(k-j+1)$. Thus $\alpha_j = 1/(k-j+1)$ gives the finite TailRL gradient. These weights are unnormalized; dividing by H_k would rescale the gradient. For binary rewards with success probability p, (64) becomes $-\sum_{\ell=1}^{k}(1-p)^{\ell}/\ell$, the finite MaxRL objective (Tajwar et al., 2026). The infinite population objective is recovered only in the limit under the relevant convergence conditions.

The centered estimators also agree algebraically. On a sorted batch, set $R_{(0:N)} = 0$ and define TailRL's weights

$$\omega_{(t)} = \sum_{j=1}^{t} \frac{R_{(j:N)} - R_{(j-1:N)}}{N-j+1}, \qquad \bar{\omega} = \frac{1}{N}\sum_{t=1}^{N} \omega_{(t)} = \frac{R_{(N:N)}}{N}. \tag{65}$$

For OrderGrad with $k = N-1$ and the harmonic α, subset counting gives

$$\frac{k}{N} a_i^{(\alpha)} = v^{\alpha} - v^{\alpha,(-i)} = \omega_i - \bar{\omega}, \qquad \widehat{g}_{\text{LR-OG}}^{\alpha} = \sum_i (\omega_i - \bar{\omega}) S_i. \tag{66}$$

To verify the first equality, partition the size-k subsets by inclusion of i. For the second, adjacent sorted coefficients on both sides differ by $(R_{(t+1:N)} - R_{(t:N)})/(N-t)$ and both vectors sum to zero. The identity also holds with ties. It recovers centered TailRL's order-$(N-1)$ gradient; the uncentered weights target order N (Ramasubramanian et al., 2026).

For binary rewards, let $C = \sum_i R_i$. Equation (66) reduces to

$$\widehat{g}_{\text{LR-OG}}^{\alpha} = \begin{cases} \displaystyle\sum_{i=1}^{N} \left(\frac{R_i}{C} - \frac{1}{N}\right) S_i, & C > 0, \\ 0, & C = 0. \end{cases} \tag{67}$$

This is the centered on-policy MaxRL update, including its all-failure convention (Tajwar et al., 2026, Algorithm 1). The equality uses the displayed sum-of-scores scaling; an extra mean reduction rescales the update. It concerns the on-policy gradient before PPO clipping or further advantage normalization. MaxPO is a different specialization: its canonical Max@k advantage uses only $\alpha_k = 1$, whereas MaxRL and TailRL use the harmonic vector.

I Additional Toy Illustrative Experiments

I.1 Illustrative Experiment on Portfolio Management using CVaR

High-yield tail-risk example. We include a small synthetic trading example to make the effect of lower-tail training visually explicit. The experiment uses the static `synthetic_tailrisk` environment. Each episode contains three risky assets and a cash asset. The risky assets are named

`high_yield_tailrisk`, `defensive_hedge`, `balanced`.

The cash asset has zero return. Its portfolio weight is capped at 10%, so that a lower-tail objective cannot solve the problem by moving entirely to cash. Policies are long-only and fully invested:

$$w_{t,a} \geq 0, \qquad \sum_a w_{t,a} = 1, \qquad w_{t,\text{cash}} \leq 0.10.$$

The risky-asset returns are generated by a simple one-factor negative-skew model. In normal periods,

$$r_{t,a} = \frac{\mu_a}{252} + b_a f_t + \frac{\sigma_a}{\sqrt{252}}\epsilon_{t,a}, \qquad f_t \sim \mathcal{N}(0, 0.004^2), \qquad \epsilon_{t,a} \sim \mathcal{N}(0,1),$$

with annualized drift parameters

$$\mu = (0.70,\ 0.030,\ 0.095),$$

annualized volatility parameters

$$\sigma = (0.11,\ 0.065,\ 0.075),$$

and factor loadings

$$b = (0.80,\ -0.10,\ 0.25).$$

Thus, the first asset has very high normal-period drift and positive factor exposure, the second asset is a defensive hedge with negative factor loading, and the third asset has intermediate behavior.

With probability p per episode, an unobserved disaster occurs at a random time. If an event occurs, the event day is sampled uniformly from the eligible post-lookback portion of the simulated tensor. In the reported horizons this corresponds to total-index days 20–97 in the 100-day training tensor and days 36–177 in the 180-day deployment tensor, where the first 20 days are used as the lookback window. At the event, the high-yield asset receives a large negative jump,

$$J \sim \mathrm{Unif}(0.50, 0.70), \qquad r_{\tau,\mathrm{tail}} \leftarrow r_{\tau,\mathrm{tail}} - J.$$

The hedge receives a positive jump,

$$H \sim \mathrm{Unif}(0.12, 0.26), \qquad r_{\tau,\mathrm{hedge}} \leftarrow r_{\tau,\mathrm{hedge}} + H,$$

and the balanced asset receives a smaller loss,

$$B \sim \mathrm{Unif}(0.015, 0.050), \qquad r_{\tau,\mathrm{balanced}} \leftarrow r_{\tau,\mathrm{balanced}} - B.$$

Two aftershock days follow when they remain inside the episode. On the next two days, with fractions $c_1 = 0.28$ and $c_2 = 0.14$, the high-yield asset loses $c_\ell J$, the hedge gains $0.30 c_\ell J$, and the balanced asset loses $0.07 c_\ell J$. All simple returns are clipped to the interval $[-0.82, 0.35]$.

The training distribution uses

$$p_{\mathrm{train}} = 0.10,$$

whereas the main deployment distribution used for Figure 13 and Table 2 uses

$$p_{\mathrm{test}} = 0.35.$$

The launch script also evaluates a post-training stress grid

$$p \in \{0.00, 0.10, 0.20, 0.35, 0.50, 0.65\},$$

but the compact publication figure and table report the main deployment setting $p_{\mathrm{test}} = 0.35$. No contiguous crisis block is used in this static tail-risk figure; the deployment shift is the higher per-episode disaster probability.

Policy, information set, and wealth dynamics. The policy observes only causal features. At each rebalance date, its input contains the trailing mean and volatility of each risky asset over a 20-day lookback window, the previous portfolio weights, the most recent realized portfolio return, an exponential moving average of portfolio return, current drawdown, and normalized time. The rolling means are multiplied by 252, the rolling volatilities by $\sqrt{252}$, and the last and exponential-moving-average portfolio returns by 25. With three risky assets and cash, this gives a 14-dimensional feature vector:

$$2 \times 3 \text{ asset moments} + 4 \text{ previous weights} + 4 \text{ scalar state variables}.$$

The policy does not observe the disaster indicator or the future disaster time.

We use the default linear policy, so the feature vector is mapped directly to four portfolio logits, one for each risky asset and one for cash. The final linear layer is initialized at zero, so training starts from equal logits rather than from an arbitrary corner. During training, Gaussian action noise with

standard deviation 0.35 is added to the logits. During deployment, the deterministic mean logits are used.

If $\tilde{w}_t = \mathrm{softmax}(z_t)$ denotes the raw softmax weights, the cash cap is implemented as

$$w_{t,\mathrm{cash}} = 0.10\,\tilde{w}_{t,\mathrm{cash}}, \qquad w_{t,a} = (1 - w_{t,\mathrm{cash}})\frac{\tilde{w}_{t,a}}{\sum_{b\in\mathcal{R}} \tilde{w}_{t,b}}, \qquad a \in \mathcal{R},$$

where $\mathcal{R}$ is the set of risky assets. Thus, the final cash weight is always at most 10%, and the remaining mass is renormalized across risky assets.

Training episodes have 80 traded days and deployment rollouts have 160 traded days. The first 20 simulated days are used only to form the initial lookback window. Portfolios are rebalanced every 5 traded days. The same weight is held between rebalances. Proportional transaction costs are charged on the first day of each rebalance block with coefficient

$$\lambda = 0.001.$$

If w_t is the new weight vector and w_{t^-} is the previous weight vector, the first day of the block uses

$$W_{t+1} = W_t \left(1 + w_t^\top r_{t+1} - \lambda \|w_t - w_{t^-}\|_1\right),$$

and subsequent days in the same block use the same portfolio weight without an additional rebalance cost. The episode score is terminal log wealth,

$$Z_i(\theta) = \log W_{T,i}(\theta).$$

Methods. Both methods use the same simulator, policy class, optimizer, random-seed protocol, minibatch size, and vectorized likelihood-ratio estimator. Each training iteration simulates a fresh minibatch of synthetic episodes. The optimizer is Adam with learning rate 0.003 and gradient clipping threshold 1.0. Both methods are trained for 750 optimization steps with minibatch size

$$B = 256.$$

The REINFORCE baseline optimizes mean terminal log wealth. In the implementation the terminal log-wealth rewards are centered and normalized within the minibatch,

$$A_i^{\mathrm{mean}} = \frac{Z_i - \bar{Z}}{\widehat{\sigma}_Z + \varepsilon}, \qquad \bar{Z} = \frac{1}{B}\sum_{j=1}^{B} Z_j,$$

and the policy-gradient loss is

$$-\frac{1}{B}\sum_{i=1}^{B} A_i^{\mathrm{mean}} \sum_t \log \pi_\theta(a_{i,t} \mid s_{i,t}).$$

The normalization changes the scale of the gradient estimate but not the mean-return objective being targeted.

The robust variant replaces the mean-return advantage by an OrderGrad advantage for a lower-tail L-statistic. Let $Z_{(1)} \le \cdots \le Z_{(B)}$ denote the sorted terminal log wealths. The lower-tail target is

$$L_\alpha(Z) = \frac{1}{\lceil \alpha B \rceil} \sum_{j=1}^{\lceil \alpha B \rceil} Z_{(j)}.$$

In the reported run,

$$\alpha = 0.20, \qquad K = 128,$$

where K is the OrderGrad interpolation parameter. The implementation computes the expected OrderGrad advantage for the statistic `LowerTailMean:0.20` and then normalizes the resulting advantages by their minibatch standard deviation before applying the same REINFORCE likelihood-ratio loss.

The reported static deployment results aggregate 1280 deterministic deployment paths per method. Since each seed uses 256 deployment rollouts, this corresponds to five training seeds. The deployment evaluation is deterministic: no Gaussian action noise is added to policy logits at test time.

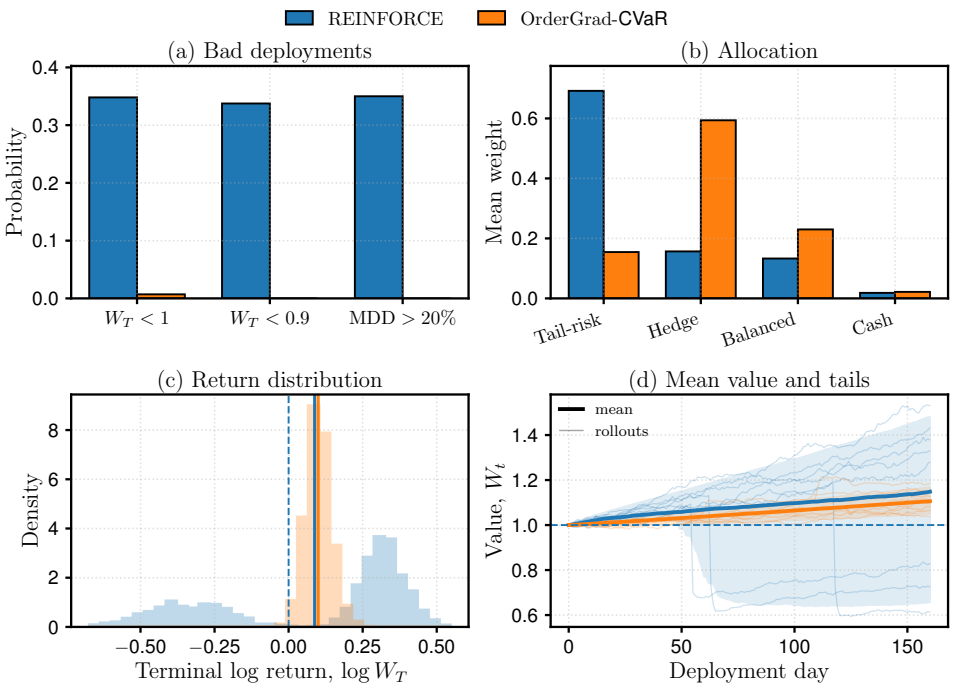


Figure 13: High-yield tail-risk trading example. Panel (a) reports bad deployment probabilities: losing money, ending below $W_T < 0.9$, and suffering a maximum drawdown larger than 20% in magnitude. Panel (b) reports average deployment portfolio weights. Panel (c) shows the distribution of terminal log returns. Panel (d) shows the mean deployment wealth, a 10–90% band, and representative individual rollouts. Standard REINFORCE attains a higher mean cumulative return by concentrating on the high-yield tail-risk asset, but this produces a broad left tail and many bad deployments. OrderGrad-CVaR reallocates toward the hedge and balanced assets, giving lower mean upside but substantially higher reliability.

Table 2: Deployment metrics in the high-yield tail-risk example. Higher is better for mean return, profitable paths. Lower is better for bad-outcome probabilities and drawdown severity.

Method	Mean return	Profitable	$W_T < 0.9$	MDD > 20%	Mean MDD
REINFORCE	14.6%	64.7%	34.2%	35.4%	−18.9%
OrderGrad-CVaR	10.7%	99.5%	0.0%	0.0%	−2.2%

Result. Figure 13 shows the main effect. Standard REINFORCE places most of its capital in the high-yield tail-risk asset, whereas OrderGrad-CVaR substantially reduces this exposure and reallocates toward the hedge and balanced assets. In the plotted run, the average high-yield exposure falls from 70.2% for REINFORCE to 16.0% for OrderGrad-CVaR, while the hedge allocation increases from 15.1% to 59.2%. The cash weights are small for both methods, 1.6% for REINFORCE and 2.2% for OrderGrad-CVaR, so the robust method is not winning by simply moving to cash.

This allocation shift produces a pronounced reliability difference. REINFORCE has the larger mean cumulative return, 14.6% versus 10.7% for OrderGrad-CVaR, because it benefits from high-yield exposure in favorable rollouts. However, the deployment distribution is much less stable: REINFORCE loses money in 35.3% of deployment paths, falls below terminal wealth $W_T < 0.9$ in 34.2% of paths, and experiences a maximum drawdown larger than 20% in magnitude in 35.4% of paths. OrderGrad-CVaR loses money in only 0.47% of paths, has no paths below $W_T < 0.9$ in this run, and has no paths exceeding the 20% drawdown threshold. The lower-tail statistics show the same pattern: the 10th percentile terminal log return improves from −0.429 for REINFORCE to 0.046 for OrderGrad-CVaR, while mean maximum drawdown improves from −18.9% to −2.2%. The daily 5% CVaR also improves from −3.21% to −0.57%.

Interpretation. The experiment is deliberately simple: it is designed to isolate the effect of the training objective rather than to model a complete financial market. The result should therefore not be read as evidence that CVaR training always gives higher average return. Instead, it demonstrates the expected risk-sensitive behavior in a controlled negative-skew allocation problem. The mean-return policy accepts the high-yield tail-risk trade because its ordinary outcomes are attractive. The CVaR-trained policy gives more weight to the bad episodes observed during training and therefore sacrifices some average upside in exchange for a substantially better lower tail. This is the behavior desired when the deployment criterion values reliability, drawdown control, or avoidance of rare large losses more than mean return alone.

I.2 Robust regression with an order-statistic training objective

We evaluate a synthetic robust-regression problem designed to test whether an order-statistic objective can suppress a coherent set of corrupted labels. Clean examples are generated from

$$y = 2.0x - 1.0 + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, 0.25^2),$$

with $x \sim \mathrm{Unif}[-3, 3]$. A fraction 0.12 of the training labels is replaced by samples from a wrong linear relation,

$$y = -1.4x + 4.2 + \epsilon_{\mathrm{out}}, \qquad \epsilon_{\mathrm{out}} \sim \mathcal{N}(0, 0.45^2).$$

The model is the linear predictor $\hat{y} = wx + b$. We compare the ordinary mean reward objective, the median reward objective, and a trimmed L-statistic objective. The per-example reward is minus squared error, so the trimmed objective emphasizes examples that are well fit by the current model and downweights both tails of the reward order statistics. In the reported run, the optimizer uses minibatches of size 256, 180 Adam steps, and the trimmed objective `TrimM:4`.

Table 3 reports clean test error averaged over 10 random seeds. The best clean MSE is obtained by Median with mean final MSE 0.0641. This demonstrates that the order-statistic objective can recover a predictor closer to the uncontaminated data-generating process than objectives that average over the corrupted population.

Figure 14 shows the fitted lines for one representative training set, illustrating that the trimmed objective follows the clean population line while the mean objective is pulled toward the corrupted population

Table 3: Robust-regression summary. Clean test MSE and parameter errors are averaged over random seeds; SE denotes the standard error of the final clean MSE.

Method	Final clean MSE	SE	$\lvert\hat{w} - w_\star\rvert$	$\lvert\hat{b} - b_\star\rvert$
Median	0.0641	0.00112	0.0139	0.0189
Trimmed ($M = 4$)	0.0656	0.00107	0.0237	0.0226
Mean	1.1	0.101	0.433	0.667

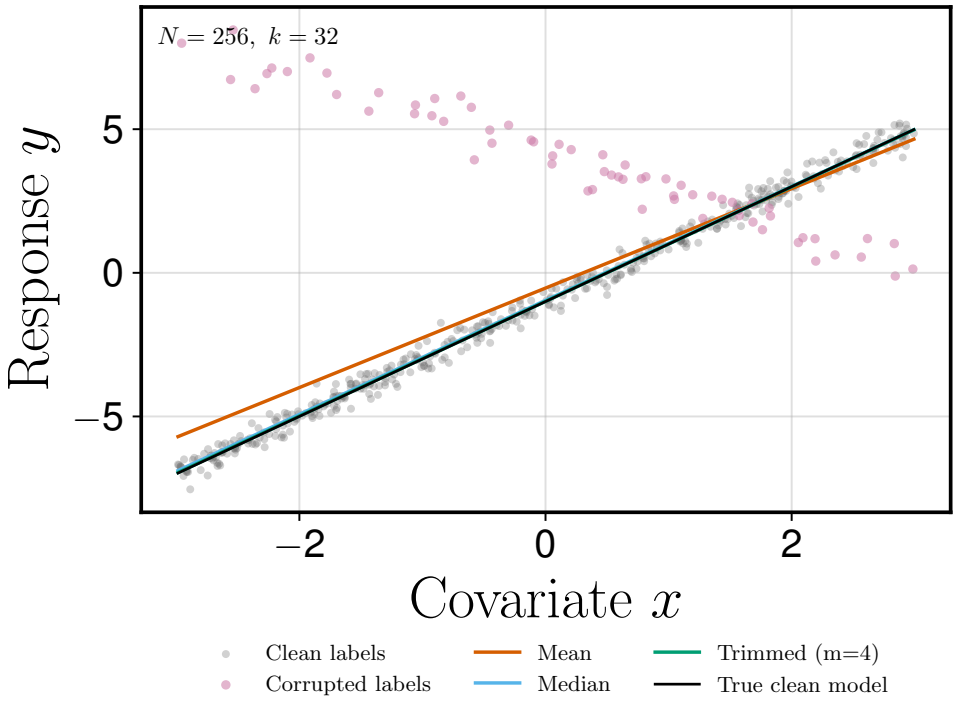


Figure 14: Representative robust-regression fit panel.

J LLM Experiments

In this section, we provide additional details and results for the LLM experiments in Sec. 5.

J.1 Setting

All models are trained with a learning rate of 10^{-6}, a batch size of 1024, and a PPO mini-batch size of 256. For each input problem, we roll out 8 responses using a temperature of 1.0. Both backbones are trained on the MATH dataset (Hendrycks et al., 2021). OrderGrad is applied prompt-conditionally: for a fixed prompt c, the rollout group is treated as samples from $p_\theta(\cdot \mid c)$ and rank advantages are computed within that group. For all methods, the advantage is normalized by the group standard deviation; this normalization, together with PPO clipping, is a practical stabilizer and is not part of the exact unbiasedness statement in the theory sections. MaxPO is swept over $K \in \{2, 4, 6\}$ and our method over $(M, K) \in \{(1, 2), (2, 4), (2, 6), (3, 6)\}$. The prompt template used is presented below.

Prompt Template (Train)

```
<|im_start|>system
You are a helpful assistant.<|im_end|>
<|im_start|>user
{input}
```

```
Please reason step by step, and put your final answer within \boxed{}.<|im_end|>
<|im_start|>assistant
```

Prompt Template (Test)

```
<|im_start|>system
Please reason step by step, and put your final answer within \boxed{}.<|im_end|>
<|im_start|>user
{input}<|im_end|>
<|im_start|>assistant
```

J.2 Additional Results

Tables 4 and 5 report the task-average pass@k on Qwen2.5-Math-7B and Qwen3-4B-Base, respectively. Table 6 reports the per-benchmark pass@1 and pass@256. Figures 15 and 16 provide the per-benchmark pass@k curves.

Table 4: Task-average pass@k on Qwen2.5-Math-7B.

Method	k=1	k=2	k=4	k=8	k=16	k=32	k=64	k=128	k=256
base	0.224	0.319	0.415	0.502	0.575	0.633	0.682	0.728	0.772
GRPO	**0.423**	0.485	0.537	0.585	0.629	0.669	0.706	0.742	0.776
MaxPO (K=2)	0.419	**0.491**	**0.550**	**0.599**	0.643	0.682	0.717	0.749	0.779
MaxPO (K=4)	0.388	0.466	0.531	0.587	0.635	0.678	0.719	0.760	**0.802**
MaxPO (K=6)	0.331	0.427	0.507	0.573	0.626	0.672	0.715	0.755	0.793
Ours (Top-1@2)	0.422	0.491	0.547	0.598	0.642	0.683	0.720	0.756	0.792
Ours (Top-2@4)	0.406	0.481	0.544	0.598	**0.643**	0.683	0.720	0.757	0.798
Ours (Top-2@6)	0.399	0.476	0.539	0.594	0.641	**0.683**	**0.723**	**0.762**	0.802
Ours (Top-3@6)	0.393	0.469	0.530	0.582	0.630	0.674	0.715	0.754	0.790

Table 5: Task-average pass@k on Qwen3-4B-Base.

Method	k=1	k=2	k=4	k=8	k=16	k=32	k=64	k=128	k=256
base	0.310	0.399	0.475	0.540	0.594	0.643	0.688	0.729	0.764
GRPO	0.411	0.466	0.509	0.544	0.578	0.613	0.646	0.680	0.719
MaxPO (K=2)	0.403	0.474	0.531	0.581	0.628	0.674	0.714	0.749	0.781
MaxPO (K=4)	0.359	0.453	0.522	0.575	0.621	0.665	0.710	0.750	0.785
MaxPO (K=6)	0.319	0.422	0.500	0.560	0.610	0.655	0.696	0.734	0.768
Ours (Top-1@2)	0.389	0.460	0.520	0.575	0.627	0.675	0.715	0.751	0.785
Ours (Top-2@4)	**0.415**	**0.494**	**0.555**	**0.607**	**0.656**	**0.702**	**0.742**	**0.778**	0.810
Ours (Top-2@6)	0.370	0.461	0.531	0.589	0.642	0.692	0.738	0.777	**0.812**
Ours (Top-3@6)	0.378	0.475	0.545	0.601	0.652	0.698	0.739	0.772	0.802

Table 6: Per-benchmark pass@1 / pass@256 for both backbones.

Method	AIME24	AIME25	AMC23	MATH500	Minerva	Avg.
Qwen2.5-Math-7B						
base	0.118 / 0.705	0.051 / 0.466	0.365 / 0.990	0.468 / 0.959	0.117 / 0.741	0.224 / 0.772
GRPO	0.298 / 0.704	0.090 / 0.488	0.617 / 0.988	0.747 / 0.947	**0.361** / 0.753	**0.423** / 0.776
MaxPO (K=2)	**0.307** / 0.723	**0.105** / 0.465	**0.637** / 0.980	**0.747** / 0.958	0.300 / 0.771	0.419 / 0.779
MaxPO (K=4)	0.267 / 0.743	0.084 / **0.557**	0.607 / 0.982	0.728 / 0.958	0.254 / 0.772	0.388 / **0.802**
MaxPO (K=6)	0.208 / 0.717	0.075 / 0.521	0.537 / **0.998**	0.645 / **0.959**	0.193 / 0.768	0.331 / 0.793
Ours (Top-1@2)	0.291 / **0.759**	0.093 / 0.517	0.629 / 0.974	**0.747** / 0.952	0.351 / 0.759	0.422 / 0.792
Ours (Top-2@4)	0.283 / 0.758	0.095 / 0.521	0.624 / 0.983	0.739 / 0.957	0.290 / 0.772	0.406 / 0.798
Ours (Top-2@6)	0.276 / 0.756	0.085 / 0.521	0.613 / 0.985	0.727 / 0.956	0.292 / **0.789**	0.399 / 0.802
Ours (Top-3@6)	0.268 / 0.741	0.089 / 0.482	0.607 / 0.990	0.723 / 0.955	0.278 / 0.783	0.393 / 0.790
Qwen3-4B-Base						
base	0.102 / 0.606	0.072 / 0.472	0.454 / 0.990	0.674 / 0.961	0.249 / 0.792	0.310 / 0.764
GRPO	0.164 / 0.604	0.105 / 0.377	0.583 / 0.966	**0.788** / 0.933	**0.416** / 0.713	0.411 / 0.719
MaxPO (K=2)	0.165 / 0.660	0.103 / 0.519	0.584 / 0.996	0.780 / 0.960	0.379 / 0.772	0.403 / 0.781
MaxPO (K=4)	0.134 / 0.648	0.122 / 0.564	0.504 / 0.991	0.713 / 0.967	0.319 / 0.753	0.359 / 0.785
MaxPO (K=6)	0.108 / 0.589	0.094 / 0.548	0.434 / 0.976	0.664 / 0.966	0.295 / 0.762	0.319 / 0.768
Ours (Top-1@2)	0.148 / 0.656	0.092 / 0.563	0.571 / **0.996**	0.771 / 0.958	0.364 / 0.753	0.389 / 0.785
Ours (Top-2@4)	**0.189** / **0.760**	**0.151** / 0.563	**0.590** / 0.989	0.783 / **0.968**	0.364 / 0.773	**0.415** / 0.810
Ours (Top-2@6)	0.160 / 0.747	0.097 / **0.574**	0.538 / 0.992	0.731 / 0.967	0.323 / **0.778**	0.370 / **0.812**
Ours (Top-3@6)	0.170 / 0.732	0.123 / 0.557	0.532 / 0.991	0.735 / 0.964	0.330 / 0.766	0.378 / 0.802

Table 7: Task-average pass@k on Qwen2.5-Math-7B with a response-length reward.

Method	k=1	k=2	k=4	k=8	k=16	k=32	k=64	k=128	k=256	k=512	k=1024
Base	0.224	0.319	0.415	0.502	0.575	0.633	0.682	0.728	0.772	0.817	0.859
GRPO	0.423	0.485	0.537	0.585	0.629	0.669	0.706	0.742	0.776	0.809	0.841
Ours (Top-2@4)	0.406	0.481	0.544	0.598	0.643	0.683	0.720	0.757	0.798	0.843	0.887
GRPO w/ length penalty	0.157	0.191	0.222	0.249	0.270	0.289	0.305	0.318	0.331	0.345	0.363
Ours (Top-2@4, Bottom-2@4)	0.403	0.470	0.525	0.571	0.614	0.656	0.698	0.738	0.778	0.821	0.867

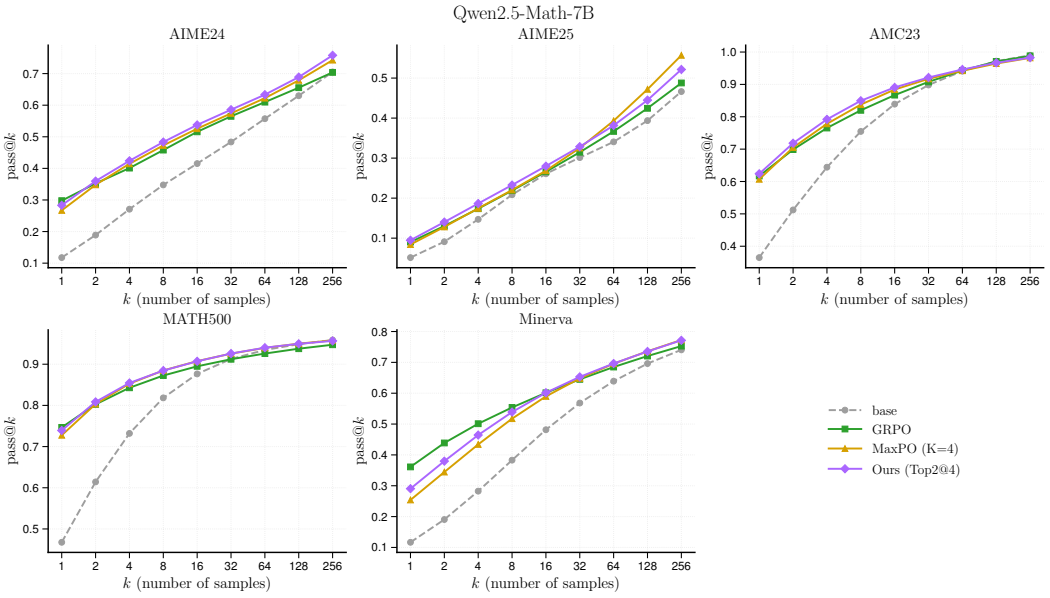


Figure 15: Per-task pass@k on Qwen2.5-Math-7B.

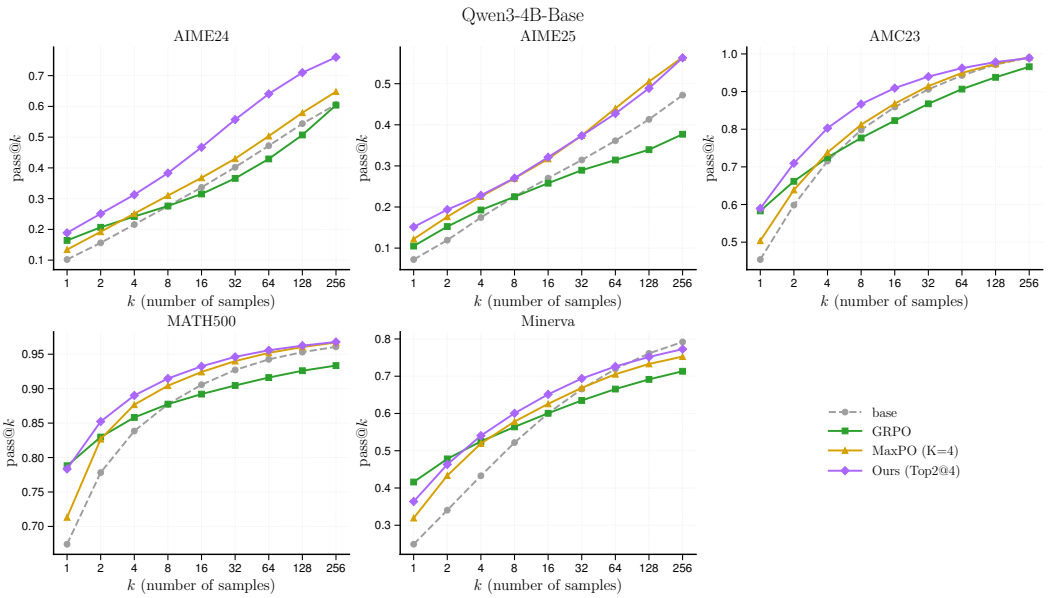


Figure 16: Per-task pass@k on Qwen3-4B-Base.

Table 7 shows the pass@k results for the length penalty on Qwen2.5-Math-7B. Figure 17 provides the per-benchmark pass@k curves.

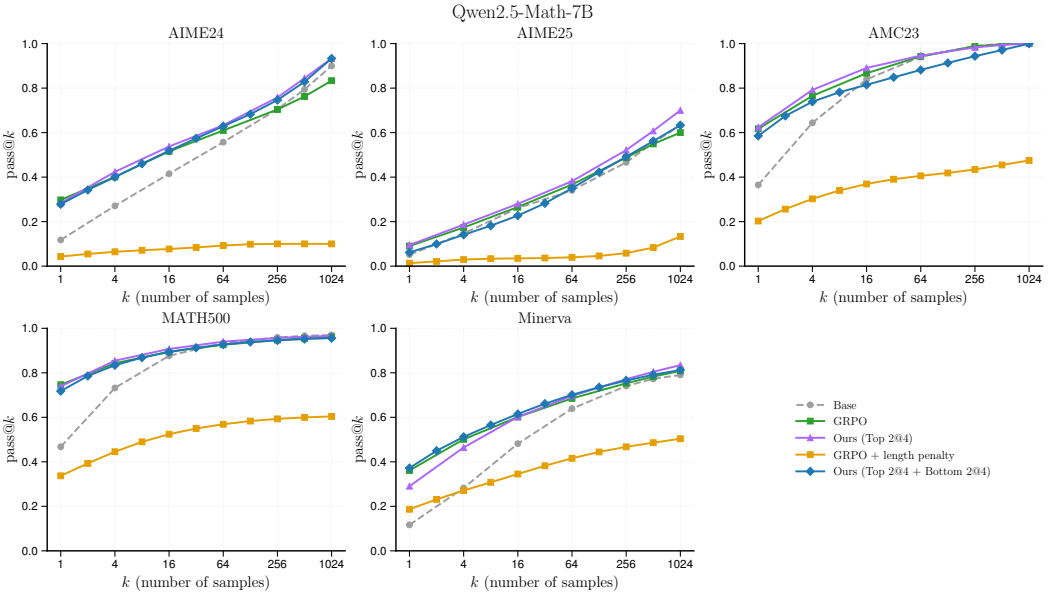


Figure 17: Per-benchmark pass@k curves on Qwen2.5-Math-7B with response length penalty. Top-2@4 by correctness reward; Bottom-2@4 by response-length reward.

Algorithm 1 OrderGrad PPO

1: **repeat**
2: Collect trajectories under π_θ and compute returns R_t^λ.
3: For each (s_t, a_t), form $q_t \leftarrow Q_\phi(s_t, \cdot)$ and $\pi_t \leftarrow \pi_\theta(s_t)$.
4: Construct $\tilde{q}_t$ by replacing the entry of q_t corresponding to a_t with R_t^λ.
5: Compute the advantage $A(t)$ following Eq. 33 with $\tilde{q}_t$ and π_t for the chosen order-statistic objective.
6: Update the actor by the PPO surrogate in Eq. 68 with $A(t)$; update the critic Q_ϕ toward R_t^λ.
7: **until** convergence

K RL Experiments

In this appendix, we describe an on-policy RL algorithm based on OrderGrad in the known-$(\bar{R}, p)$ regime explained in App. C, and report experimental results with MinAtar (Young & Tian, 2019). The experiment closely follows the setup of Nishimori et al. (2026).

PPO surrogate. PPO collects trajectories under the behavior policy π_{old}, computes λ-returns R_t^λ, and forms advantages $A(t) = R_t^\lambda - V_\phi(s_t)$. It then optimizes a clipped, importance-weighted surrogate with $r_\theta(t) = \pi_\theta(a_t \mid s_t)/\pi_{\text{old}}(a_t \mid s_t)$ and clipping parameter $\epsilon > 0$:

$$\mathcal{L}_{\text{PPO}}(\theta) := \mathbb{E}\min\Big(r_\theta(t)A(t),\ \text{clip}\big(r_\theta(t), 1-\epsilon, 1+\epsilon\big)A(t)\Big). \tag{68}$$

OrderGrad PPO uses the same surrogate, but replaces the standard value-based advantage with the order-statistic advantage described below.

OrderGrad PPO. We instantiate OrderGrad in a PPO-style algorithm, which we call OrderGrad PPO. The main difference from standard PPO is that the policy advantage is computed from an order-statistic objective using a Q-critic Q_ϕ, where ϕ denotes the parameters of the Q-network, rather than from the usual value-based advantage estimate used as a proxy for the return. Specifically, for each state-action pair (s_t, a_t), we first form the vector $q_t = Q_\phi(s_t, \cdot) \in \mathbb{R}^{|\mathcal{A}|}$ over the action set $\mathcal{A}$ and the policy vector $\pi_t = \pi_\theta(s_t) \in \Delta^{|\mathcal{A}|}$. We then construct $\tilde{q}_t$ by replacing the entry of q_t corresponding to the sampled action a_t with the trajectory return R_t^λ, following Nishimori et al. (2026), to fully use the on-policy return information. The advantage is computed following Eq. 33 with $\tilde{q}_t$ and π_t for the chosen order-statistic objective, such as Top-M@K or Bottom-M@K. This is a practical actor-critic instantiation rather than a direct implementation of the exact finite-action estimator: it uses a learned Q-critic, substitutes a sampled return into one Q entry, and optimizes a clipped PPO surrogate. Algorithm 1 summarizes the resulting procedure. At each iteration, we collect trajectories with the current policy, compute λ-returns, estimate order-statistic improvement scores from the Q-critic Q_ϕ and policy π_θ, and use the resulting advantages in the PPO surrogate in Eq. 68.

Setup. We evaluate this algorithm on MinAtar environments implemented in pgx (Young & Tian, 2019; Koyamada et al., 2023a), using Asterix, Breakout, and Space Invaders, following Nishimori et al. (2026). The goal of this experiment is to test whether OrderGrad can modulate exploration behavior through its order-statistic parameters, without relying on explicit entropy regularization. We use the Top-M@K objective as a representative order-statistic objective of OrderGrad, fix $K = 10$, and vary M. We compare against PPO-V and PPO-Q, both with and without entropy regularization. All methods are trained for 10^7 environment steps over 10 random seeds. PPO hyperparameters mostly follow the pgx defaults, while OrderGrad PPO uses the same setup except that the GAE parameter λ is tuned separately. We report the median, IQM, and mean of the normalized score following Agarwal et al. (2021). The maximum scores used for normalization are taken from Nishimori et al. (2026): 251.15 for Breakout, 64.95 for Asterix, and 880.91 for Space Invaders.

Network architecture. We use the same network architecture as the public implementation of PPO (Koyamada et al., 2023a). For PPO-based methods, the network first maps the observation to a latent state using a shared CNN with a 2×2 convolution, ReLU activation, and average pooling,

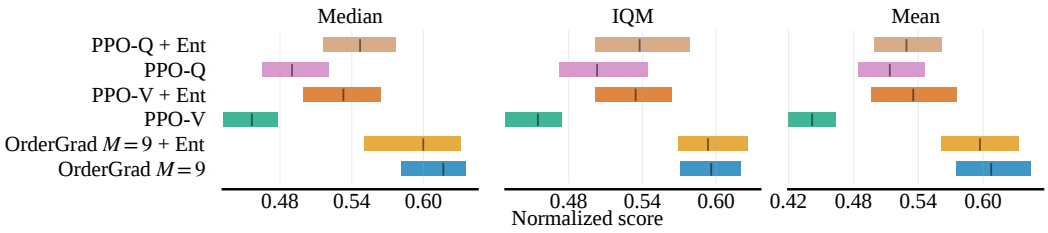


Figure 18: Aggregate MinAtar performance. We compare OrderGrad PPO with $M = 9$ against PPO-V and PPO-Q, with and without entropy regularization. Scores are normalized and aggregated across environments.

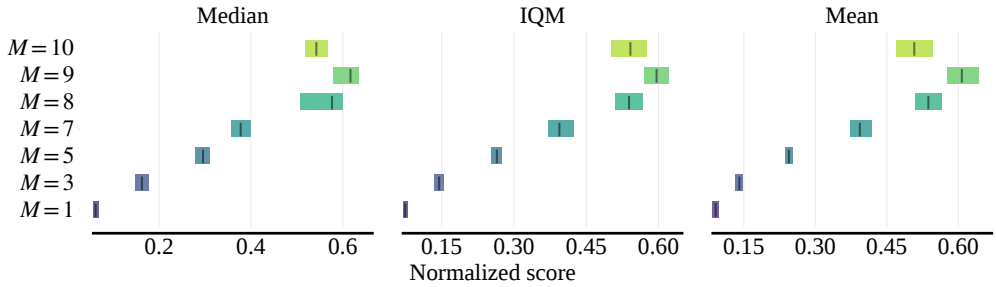


Figure 19: Effect of M on MinAtar without entropy regularization. We report aggregate normalized evaluation return across games. All OrderGrad curves use entropy coefficient 0.0. The best performance occurs around $M = 9$.

followed by an MLP. The latent representation then branches into an actor head and a critic head. The actor head consists of two hidden layers with ReLU and Tanh activations and outputs action logits. For OrderGrad PPO and PPO-Q, the critic head consists of two hidden layers and outputs per-action Q-values, yielding a Q-critic $Q_\phi(s, \cdot)$. For PPO-V, we use the corresponding scalar value critic.

Results. Figure 18 shows the aggregate MinAtar performance computed with rliable (Agarwal et al., 2021). In these runs, OrderGrad PPO with $M = 9$ outperforms the PPO baselines in terms of normalized score. We further analyze the effect of the parameter M in Fig. 19. The best aggregate performance is obtained around $M = 9$, suggesting that the order-statistic parameter controls not only the objective but also the practical exploration–exploitation behavior of the learned policy. This interpretation is supported by the entropy curves in Fig. 20, where all OrderGrad runs use entropy coefficient 0.0. Smaller values of M slow entropy decay and maintain exploration for longer, whereas larger values of M lead to faster entropy decay. Since this variant uses critic approximation and PPO clipping, these results should be read as an empirical instantiation of the OrderGrad idea rather than a separate unbiasedness result. Overall, the results support the flexibility of OrderGrad when combined with practical deep policy-gradient algorithms.

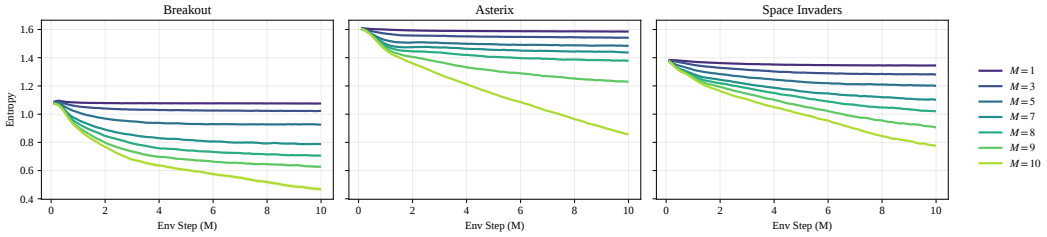


Figure 20: Policy entropy under different values of M without entropy regularization. All curves use entropy coefficient 0.0. Smaller M slows entropy decay and encourages more persistent exploration, whereas larger M leads to faster entropy decay.

Table 8: Hyperparameters for PPO-V/PPO-Q.

Name (symbol)	**Value**
total timesteps	1.0×10^7
learning rate	3.0×10^{-4}
rollout length	128
parallel envs	1024
update epochs	3
minibatch size	1024
discount γ	0.99
GAE λ	0.95
clip ϵ	0.2
entropy coefficient	$(0.0, 0.01)$
value loss coefficient	0.5
max grad norm	0.5
optimizer	Adam with global-norm clip

Table 9: Hyperparameters for OrderGrad PPO.

Name (symbol)	**Value**
total timesteps	1.0×10^7
learning rate	3.0×10^{-4}
rollout length	128
parallel envs	1024
update epochs	3
minibatch size	1024
discount γ	0.99
GAE λ	0.8
clip ϵ	0.2
entropy coefficient	$(0.0, 0.01)$
value loss coefficient	0.5
max grad norm	0.5
optimizer	Adam with global-norm clip
top-M draws M	$(1, 3, 5, 7, 8, 9, 10)$