Training Agent to Scale Inference-Time Reasoning

Yunzhe Qi[1,2*†] **Sirui Chen**[1*†] **Jiaru Zou**[3†] **Yanjun Zhao**[1] **Mengting Ai**[1] **Jingrui He**[1]
[1]University of Illinois Urbana-Champaign [2]Meta [3]Stanford University
{yunzheq2, jingrui}@illinois.edu

Abstract

Inference-time scaling provides a promising path for improving agentic reasoning performance, and one widely adopted form is *sample-then-select*: first sample a pool of candidate reasoning trajectories, then select from the resulting candidates. While this keeps scaling lightweight without updating the policy, it exposes a training-deployment mismatch. Standard trajectory-wise Reinforcement Learning (RL) objectives optimize normalized signals for individual trajectories, whereas choosing from a large candidate pool with the policy's own selection signal is inherently set-conditional: it requires calibrated ranking and confidence aggregation across correlated trajectories. To address this challenge, we propose AUTOPORTFOLIO, a set-conditional policy alignment framework for efficient and effective *sample-then-select* scaling. AUTOPORTFOLIO builds a tractable environment-informed target on the sampled candidates, and uses its induced Plackett-Luce (PL) *ranking* to calibrate the policy's ranking of trajectories within that set. It trains the policy with a balanced PL-motivated objective that reallocates probability mass among competing trajectories, preserving reward-aligned alternatives while sharpening away from low-quality distractors. Aligned with our policy training, our lightweight inference-time Mass Aggregation scaling adaptively perceives confidence from candidate trajectories and selects the most promising answer(s). Theoretically, we show that reducing our set-conditional loss narrows the discrepancy between policy-induced and environment-informed PL rankings on the sampled set. Empirically, across 12 challenging reasoning benchmarks, AUTOPORTFOLIO improves over strong agentic RL baselines, with the clearest gains in larger candidate pools where the set-level selection is crucial.

1 Introduction

Reasoning-centric LLM agents have made rapid progress on challenging tasks, such as mathematical reasoning [46, 37]. Beyond scaling training data and parametric capacity, recent advances highlight a complementary lever: *inference-time scaling* [28, 50, 63, 52]. By scaling up test-time computation (e.g., sampling multiple candidate trajectories), we can improve reliability on complex tasks without updating model parameters. This helps shift the burden from the model's fixed parametric capability to *how compute is used at inference*, trading adaptive test-time computation for better performance.

Among the many ways to spend test-time compute, an efficient and effective regime is *sample-then-select* inference: the system samples N candidate reasoning trajectories, then applies reranking signals, such as the policy's confidence, to decide what to keep. This turns parallel sampling into scalable gains without changing the base policy, relying only on lightweight

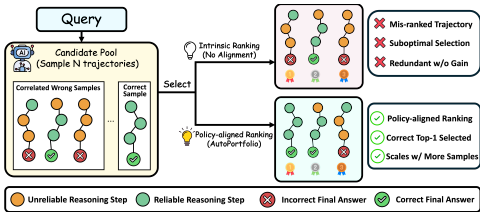

Figure 1: *Sample-then-select* inference: larger N can reveal correct answers but also correlated distractors; pointwise scoring can mis-rank redundant failures, while AUTOPORTFOLIO's policy-aligned within-set ranking surfaces the correct answer.

*Joint first authors.
†Core contributor.

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

post-hoc selection. This pattern underlies repeated-sampling approaches, including self-consistency [58] and Best-of-N/reranking inference [31, 4]. It is also crucial in practical resource-limited pipelines: coding systems can generate large pools of candidate code artifacts and filter them down to a small submission or validation *subset* [30, 64, 39], while AI-for-science workflows filter large sets of candidate drugs to a small *subset* for costly experimental validation [62, 53].

Across the *sample-then-select* scenarios, one critical challenge is turning sampled pools into reliable decisions. This requires two capabilities: (1) *generation*, which places strong candidates in the sampled pool, and (2) *within-set alignment*, which ranks stronger candidates above others in that same pool. To improve *generation* quality, standard Reinforcement Learning (RL) objectives, including group-normalized methods such as GRPO and its variants (e.g., 22, 81, 72, 20), train on trajectory-level reward signals, effectively treating candidates as separately weighted trajectories. However, the sampled groups in these standard RL formulations mainly serve to normalize trajectory-level updates, e.g., through relative baselines or advantage weights for variance or bias control [18], rather than explicitly calibrating candidate *rankings* within the pool. This creates a practical training-inference misalignment: when selection relies on the policy's inherent signal for *sample-then-select* scaling, reliable within-pool ranking becomes an intrinsic bottleneck as test-time compute scales (i.e., *within-set alignment*) [4], while standard RL objectives remain centered on individual trajectory quality (i.e., *generation*).

This misalignment is amplified when sampled trajectories are highly correlated: agentic models can latch onto a few high-likelihood reasoning patterns, yielding redundant rollouts and near-duplicate failure modes [34, 73]. In this case, increasing the scaling budget N has two competing effects: it can improve coverage by raising the chance that a strong candidate appears, but it can also make selection harder when the expanded candidate pool is dominated by high-probability distractors (Fig. 1). Thus, to leverage the policy's inherent signal for *sample-then-select* deployment, policy training requires calibrated *ranking* supervision over each sampled pool, so that the probabilities assigned by the policy can naturally induce a ranking aligned with candidates' relative quality. Motivated by this need, we close the gap by optimizing *generation* and *within-set alignment* simultaneously, internalizing the ranking logic during agentic policy training to match the *sample-then-select* scaling scenario.

Proposed Framework. We introduce AUTOPORTFOLIO, a set-conditional policy alignment framework that makes *within-set ranking* an explicit optimization target, improving both candidate generation and ranking alignment. During training, AUTOPORTFOLIO converts reward feedback on the sampled pool into a target distribution over candidates, and aligns the policy to this target by matching the induced Plackett-Luce (PL) ranking distribution on the candidate set. Instead of improving each trajectory in isolation, our balanced PL-motivated Cross-Entropy (PL-CE) objective reallocates probability mass within each candidate set, preserving reward-aligned modes while sharpening toward strong candidates. Aligned with policy training, our lightweight yet effective Mass Aggregation scaling combines trajectory confidence at the answer level, balancing broad support with peak confidence to make final selection more reliable under duplicate rollouts and brittle max-probability choices.

Theoretical and Empirical Insights. Theoretically, we show that our set-conditional objective is a principled surrogate for minimizing the discrepancy between the policy-induced PL ranking and the environment-informed optimal PL ranking. Our analysis further characterizes the induced push-pull gradients, which reallocate probability mass within the candidate set by pulling it toward high-quality candidates and pushing it away from low-quality distractors, while preserving other promising trajectories. Empirically, we evaluate AUTOPORTFOLIO on 12 challenging benchmarks, including AIME24/25, Omni-MATH, and GPQA-Diamond. Comprehensive results show that AUTOPORTFOLIO consistently outperforms strong agentic reasoning baselines, with the largest gains appearing when selection quality becomes the bottleneck in larger candidate pools.

2 Problem Setup: Sample-Then-Select Inference-Time Scaling

Reasoning Trajectories. We view agentic reasoning as a sequential decision process. A policy $\pi_\theta(\tau|\boldsymbol{x})$, with trainable parameters θ, generates a trajectory τ composed of intermediate thought steps (e.g., chain-of-thought traces, tool invocations) resulting in a final answer for a query $\boldsymbol{x}$. Consequently, the correctness of the answer hinges on the quality of the entire reasoning trajectory τ. Each trajectory induces a final answer and can be evaluated through answer-level reward.

Sample-Then-Select Inference with Reasoning Portfolio. At inference time, we improve performance by scaling up generation and then selecting from the resulting candidates. The agent first

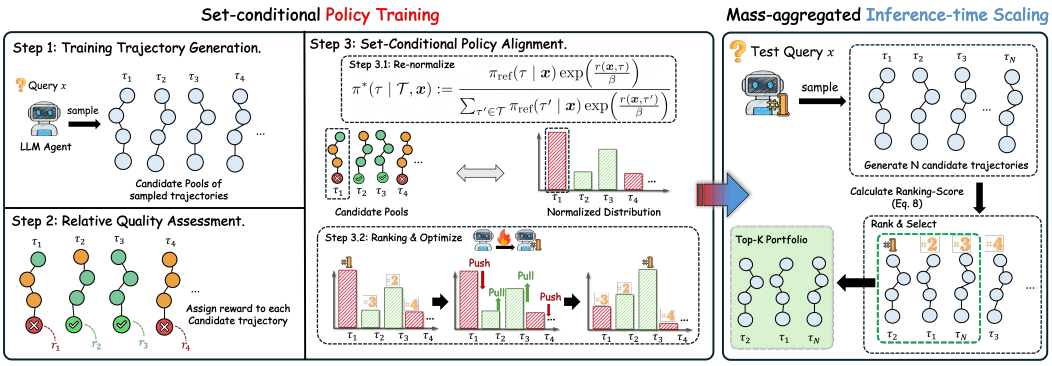

Figure 2: **AUTOPORTFOLIO pipeline.** ***Set-conditional Policy Training (left):*** for query x, the agent samples a candidate pool of reasoning trajectories, assigns a reward to each trajectory, then performs set-conditional alignment by re-normalizing the induced distribution on the pool and optimizing a listwise ranking objective that *pushes* probability mass away from over-allocated candidates and *pulls* it toward under-allocated ones (mass is reallocated within the set). ***Mass-aggregated Inference-time Scaling (right):*** given a test query, the agent samples N trajectories, computes ranking scores, and selects a reasoning portfolio for downstream use.

samples a *candidate set* $\mathcal{T}_N = \{\tau_1, \ldots, \tau_N\}$ of size N. From this set, it then applies a selection rule to produce a *reasoning portfolio* containing either one final answer ($K = 1$) or at most $K > 1$ retained answers. This formulation subsumes both *single-answer* reranking and settings where *multiple answers* are retained for downstream verification or execution [62, 53, 30, 64, 39].

Learning Objective. Each candidate trajectory τ is scored by its *induced answer*, with reward $r(\boldsymbol{x}, \tau) \in \mathbb{R}$ (e.g., F1 score). As $\mathcal{T}_N$ induces candidate answers, selecting an answer can be represented by keeping the trajectory that produces it. We therefore optimize the quality of the selected portfolio by maximizing

$$\mathbb{E}_{\mathcal{T}_N \sim \pi_\theta(\cdot|\boldsymbol{x})} \left[\sum_{\tau \in \text{Select-K}(\mathcal{T}_N; \pi_\theta)} r(\boldsymbol{x}, \tau) \right], \tag{1}$$

where Select-K$(\mathcal{T}_N; \pi_\theta) \subseteq \mathcal{T}_N$ denotes selected trajectories with distinct induced answers and cardinality at most K. This recovers standard generation when $N = K = 1$ and generalizes to *sample-then-select* scaling when $N > K \geq 1$. N controls the sampled candidate set, while K defines how many distinct induced answers are retained. Maximizing this objective requires two capabilities: (1) *Generation:* The sampled set $\mathcal{T}_N$ should contain trajectories that lead to high-quality answers. (2) *Within-set Alignment:* Given $\mathcal{T}_N$, the policy π_θ should assign higher support to trajectories that induce better answers, so that the selected candidates favor higher-quality answers.

3 Proposed Framework: AUTOPORTFOLIO

We aim to *align training with inference-time deployment*: the model is trained not only to generate strong trajectories, but also to rank the sampled pool for reliable selection. As shown in Fig. 2, AUTOPORTFOLIO realizes this through two coupled components: *(1) Set-conditional Policy Training*, which aligns the policy to environment-informed within-set rankings; *(2) Mass-aggregated Inference-time Scaling*, which aggregates trajectory confidence at the answer level to select the final portfolio.

(1) Set-conditional Policy Training (Subsec. 3.1, Algo. 1). We update the agentic policy π_θ by

- **Step 1: Training Trajectory Generation.** For each query $\boldsymbol{x}$ in a training batch, the agent samples a candidate trajectory set $\mathcal{T} := \{\tau_i\}_{i=1}^{|\mathcal{T}|} \sim \pi_\theta(\cdot|\boldsymbol{x})$, so training uses trajectories the current policy actually produces, rather than relying only on offline traces.
- **Step 2: Relative Quality Assessment.** Each trajectory τ is scored through its induced answer, $r(\boldsymbol{x}, \tau)$. Instead of using rewards pointwise, we convert them into a *set-conditional target distribution* over $\mathcal{T}$ that encodes the desired *within-set* ranking.
- **Step 3: Set-Conditional Policy Alignment (Subsecs. 3.1.1-3.1.2).** We train π_θ to match this ideal ranking using a balanced *Plackett-Luce motivated Cross-Entropy (PL-CE)* loss, leading to a

set-wise "push-pull" dynamic: *Push:* Probability mass is explicitly pushed away from low-quality candidates in the set. *Pull:* Probability mass is pulled toward high-quality candidates.

Crucially, PL-CE balances mode coverage and mode concentration: it preserves multiple reward-aligned reasoning modes while sharpening probability toward the strongest candidates.

(2) Inference-time Scaling (Subsec. 3.2, Algo. 2). To leverage aligned internal ranking of π_θ for *sample-then-select inference*, we propose a lightweight *Mass Aggregation* strategy. By aggregating the learned probabilities at the answer level, the agent can adaptively self-select the most promising answer(s) from candidates, directly optimizing the final selected set under a fixed inference budget.

3.1 Agent Training via Set-conditional Alignment

Reasoning Trajectory Rollout. Recall that for an input query $\boldsymbol{x}$, we sample trajectories (including chain-of-thought traces, tool invocations, etc.) $\mathcal{T} := \{\tau_i\}_{i=1}^{|\mathcal{T}|}$ directly from the current policy π_θ [15, 17]. Our goal is to update the policy to probabilistically prioritize high-quality trajectories.

KL-regularized RL Problem. We consider a standard Kullback-Leibler (KL)-regularized RL objective with a reference policy π_{ref} [47, 44, 6]. For an input query $\boldsymbol{x}$ and generated reasoning trajectory τ, our objective for training the policy model π_θ is:

$$\max_{\pi_\theta} \mathbb{E}_{\tau\sim\pi_\theta(\tau|\boldsymbol{x})}\big[r(\boldsymbol{x},\tau)\big] - \beta\cdot\mathbb{D}_{\text{KL}}\big(\pi_\theta(\tau|\boldsymbol{x})\|\pi_{\text{ref}}(\tau|\boldsymbol{x})\big), \tag{2}$$

where τ refers to the generated agentic reasoning trajectory, and reward function $r(\cdot,\cdot)$ evaluates the corresponding generated answers. $\beta > 0$ controls the regularization intensity. The above RL problem leads to a closed-form solution [36, 44], such that the *optimal policy* π^* solving Eq. 2 is

$$\pi^*(\tau|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})}\cdot\pi_{\text{ref}}(\tau|\boldsymbol{x})\exp\left(\beta^{-1}r(\boldsymbol{x},\tau)\right). \tag{3}$$

The partition function $Z(\boldsymbol{x}) := \sum_{\tau'}\pi_{\text{ref}}(\tau'|\boldsymbol{x})\exp\left(\beta^{-1}r(\boldsymbol{x},\tau')\right)$ is intractable, as it ranges over *all possible* trajectories, making it infeasible to directly apply the optimal policy π^* for decision making.

Turning *Environment* into Optimal Policy π^*

Eq. 3 can be viewed as a principled way to *transform* an unknown environment, accessed only through rewards $r(\boldsymbol{x},\tau)$, into an explicit *target distribution* over trajectories. This target defines an optimal policy π^* that upweights high-quality trajectories while remaining stably close to the reference policy π_{ref}. Our goal is to train π_θ to align with π^*, so that high-quality reasoning trajectories become inherently more probable under the learned policy.

From Intractable π^* to Tractable Set-conditional Targets. The KL-regularized RL solution defines an *optimal* policy $\pi^*(\cdot\mid\boldsymbol{x})$ over *all possible* reasoning trajectories (Eq. 3), but its partition function $Z(\boldsymbol{x})$ is intractable. In practice, training only observes a finite rollout set $\mathcal{T} = \{\tau_i\}_{i=1}^{|\mathcal{T}|}$ with rewards. We thus convert the global target $\pi^*(\cdot\mid\boldsymbol{x})$ into a tractable, set-conditional supervision signal by restricting it to $\mathcal{T}$ and re-normalizing; on the same set, we correspondingly define the policy model's current distribution by re-normalizing $\pi_\theta(\cdot\mid\boldsymbol{x})$:

$$\pi^*(\tau\mid\mathcal{T},\boldsymbol{x}) := \frac{\pi_{\text{ref}}(\tau\mid\boldsymbol{x})\exp\left(\frac{r(\boldsymbol{x},\tau)}{\beta}\right)}{\sum_{\tau'\in\mathcal{T}}\pi_{\text{ref}}(\tau'\mid\boldsymbol{x})\exp\left(\frac{r(\boldsymbol{x},\tau')}{\beta}\right)}, \quad \pi_\theta(\tau\mid\mathcal{T},\boldsymbol{x}) := \frac{\pi_\theta(\tau\mid\boldsymbol{x})}{\sum_{\tau'\in\mathcal{T}}\pi_\theta(\tau'\mid\boldsymbol{x})}. \tag{4}$$

The first $\pi^*(\tau\mid\mathcal{T},\boldsymbol{x})$ cancels the *intractable* $Z(\boldsymbol{x})$ and specifies which trajectories the optimal policy should prefer, and by how much, *within the sampled set* $\mathcal{T}$; the second $\pi_\theta(\tau\mid\mathcal{T},\boldsymbol{x})$ captures what the current policy prefers on that same set. Together, these set-conditional distributions induce a *within-set comparison*, encouraging the policy to redistribute probability mass among the sampled candidates rather than assign absolute scores in isolation.

3.1.1 Set-conditional Loss for PL Ranking

Plackett-Luce (PL) Ranking of Reasoning Trajectories. After generating a trajectory collection $\mathcal{T}$ and evaluating the corresponding rewards, we need a principled way to *learn from their relative*

quality and *prioritize high-quality trajectories*. Thus, we formalize this using the PL model [38, 33], a standard probabilistic framework that is particularly effective in modeling ranking distributions [44, 41, 87, 5]. Given a trajectory set $\mathcal{T} = \{\tau_i\}_{i=1}^{|\mathcal{T}|}$ for query $\boldsymbol{x}$, for a permutation σ of trajectory indices $\{1, \ldots, |\mathcal{T}|\}$, we have the optimal PL model induced by the optimal π^*:

$$P_{\pi^*}(\sigma|\,\mathcal{T}, \boldsymbol{x}) := \prod_{i=1}^{|\mathcal{T}|} \frac{\exp\left(r(\boldsymbol{x}, \tau_{\sigma(i)})\right)}{\sum_{j=i}^{|\mathcal{T}|} \exp\left(r(\boldsymbol{x}, \tau_{\sigma(j)})\right)} = \prod_{i=1}^{|\mathcal{T}|} \frac{\exp\left(\beta \log \frac{\pi^*(\tau_{\sigma(i)}|\boldsymbol{x})}{\pi_{\text{ref}}(\tau_{\sigma(i)}|\boldsymbol{x})}\right)}{\sum_{j=i}^{|\mathcal{T}|} \exp\left(\beta \log \frac{\pi^*(\tau_{\sigma(j)}|\boldsymbol{x})}{\pi_{\text{ref}}(\tau_{\sigma(j)}|\boldsymbol{x})}\right)}, \tag{5}$$

where $\tau_{\sigma(i)}$ refers to the i-th ranked trajectory of permutation σ. The second equality is by transforming Eq. 3 to derive the closed-form representation of $r(\cdot, \cdot)$. Intuitively, this formulation concentrates mass on higher-reward chains while remaining anchored to the reference π_{ref}.

PL Ranking for Portfolio Learning. The PL framework is intrinsically suited to our goal of *portfolio optimization*, because it models the nuanced *relative* preference among a set of candidates. Here, our problem formulation establishes that the optimal policy π^* assigns probabilities proportional to the exponential reward (Eq. 3). This structure is naturally aligned with the PL model's exponential scoring. By adopting the PL model, we train π_θ to match this *optimal policy within each sampled set*, so that probability mass is shifted toward higher-reward trajectories relative to their competitors. This encourages the model to prioritize high-quality trajectories with correct ranking orders, rather than merely maximizing independent trajectory scores.

However, directly optimizing the discrepancy between PL models over permutations is *infeasible*, as there are factorially many possible permutations. Recall that our goal is selecting and ranking trajectories *within* a rollout set. The PL model (Eq. 5) formalizes this as a distribution over permutations σ, based on relative utilities. Thus, if $\pi_\theta(\cdot \mid \mathcal{T}, \boldsymbol{x})$ matches the target $\pi^*(\cdot \mid \mathcal{T}, \boldsymbol{x})$, the corresponding PL ranking distributions will also be close. This motivates our strategy of training π_θ to match the optimal policy π^* (Eq. 5), with the ultimate goal of achieving the optimal PL ranking P_{π^*}.

Forward CE as Set-wise Alignment towards π^*. To align the trainable policy π_θ with the target π^* on the finite set $\mathcal{T}$, a natural solution is minimizing the Cross-Entropy loss on $\mathcal{T}$. Denoting the training data $\mathcal{D}$ as a batch of query-trajectory-set pairs $(\boldsymbol{x}, \mathcal{T})$ with the corresponding rewards, we can minimize the set-conditional forward CE loss:

$$\mathcal{L}_{\text{fwd}}(\mathcal{D}) := -\frac{1}{|\mathcal{D}|} \sum\nolimits_{(\boldsymbol{x},\mathcal{T})\in\mathcal{D}} \sum\nolimits_{\tau\in\mathcal{T}} \Big[\pi^*(\tau|\mathcal{T}, \boldsymbol{x}) \log \pi_\theta(\tau|\mathcal{T}, \boldsymbol{x})\Big], \tag{6}$$

which intuitively teaches the policy to *rank and reweight* trajectories specifically to match the KL-regularized optimal policy within the generated portfolio.

Why Does Set-conditional Re-normalization Matter?

Re-normalization (Eq. 4) induces direct *mass reallocation*. Denote $s_i := \log \pi_\theta(\tau_i|\boldsymbol{x})$, $q_i = \pi^*(\tau_i|\mathcal{T}, \boldsymbol{x})$ and $p_i = \pi_\theta(\tau_i|\mathcal{T}, \boldsymbol{x})$. Using a CE loss $\mathcal{L}$ with re-normalized probabilities, it gives $\partial\mathcal{L}/\partial s_i = p_i - q_i$ with $\sum_i (p_i - q_i) = 0$: probability is pushed off over-allocated trajectories ($p_i > q_i$) and pulled onto under-allocated ones ($p_i < q_i$), preserving the total mass. This makes learning depend on *relative* probabilities within $\mathcal{T}$, which is a zero-sum mass reallocation invariant to global scaling. It directly optimizes the set ranking on $\mathcal{T}$, unlike trajectory-level advantage-weighted updates (e.g., GRPO). Derivations are in Appendix F.2.

3.1.2 Balancing Mode Covering and Mode Seeking via *Forward* & *Reverse* CE

Why Go Beyond Forward CE? Forward CE (Eq. 6) has a characteristic behavior: it encourages the learned policy to cover all trajectories that the target considers plausible. This is desirable because it maintains coverage and prevents collapse to a single reasoning pattern. However, a strong reasoning portfolio also needs sharper alignment, placing more probability on the best candidates so that the induced ranking is reliable. To balance coverage and concentration, we thus add a reverse CE term. Given query $\boldsymbol{x}$ and a rollout set $\mathcal{T}$ of reasoning trajectories, we define the set-conditional forward and reverse CE losses as follows.

(1) Forward CE is Mode Covering ("Do Not Drop Target-supported Options"). For simplicity, denote $q(\tau) = \pi^*(\tau|\mathcal{T}, \boldsymbol{x})$ and $p(\tau) = \pi_\theta(\tau|\mathcal{T}, \boldsymbol{x})$. Then, we have the forward CE loss

$$\mathcal{L}_{\text{fwd}}^{\mathcal{T}}(\pi^*, \pi_\theta) = -\sum\nolimits_{\tau\in\mathcal{T}} \big[q(\tau) \log p(\tau)\big],$$

which is *support sensitive*: if $q(\tau)$ is non-negligible but $p(\tau)$ approaches 0, then $-q(\tau)\log p(\tau)$ will be large. The forward CE loss strongly penalizes "dropping" any trajectory that the target policy assigns meaningful probability to. When π^* has multiple good modes, forward CE encourages π_θ to keep them all in play, improving overall coverage (derivations in Appendix F.1.1).

(2) Reverse CE is Mode Seeking ("Concentrate on What the Target Says is Best"). On the other hand, the reverse CE can be written as

$$\mathcal{L}^{\mathcal{T}}_{\text{rev}}(\pi^*, \pi_\theta) = -\sum\nolimits_{\tau\in\mathcal{T}} \big[p(\tau)\log q(\tau)\big].$$

Here, the loss focuses on *where the model is already placing mass*. If π_θ assigns probability to trajectories with very small $q(\tau)$, the penalty $-\log q(\tau)$ becomes large, and gradient descent rapidly reduces their probability. In contrast, increasing probability on high-q trajectories immediately reduces the loss, creating a sharpening dynamic that concentrates mass around the highest-quality modes. Reverse CE encourages concentration on the highest-reward trajectories within $\mathcal{T}$ (derivations in Appendix F.1.2).

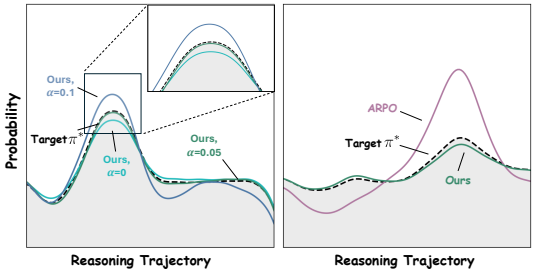

Figure 3: **Mode illustration for AIME 24.** PL-CE reallocates probability mass *within* a sampled candidate pool; tuning the reverse-CE weight α interpolates between mode-covering and mode-seeking and stays close to the target π^*. ARPO can over-concentrate on a single mode, whereas PL-CE preserves coverage while aligning better with π^*.

Putting Together: PL-CE. Averaging the loss over the training data batch $\mathcal{D}$, we define our balanced Plackett-Luce motivated CE (PL-CE) objective:

$$\mathcal{L}_{\text{PL-CE}}(\mathcal{D}) := \mathcal{L}_{\text{fwd}}(\mathcal{D}) + \alpha\,\mathcal{L}_{\text{rev}}(\mathcal{D}), \qquad (7)$$

where $\alpha \geq 0$ is a tunable coefficient. Our overall training pipeline is summarized in Algo. 1. This combined objective remains a principled surrogate for minimizing the PL ranking discrepancy.

Theorem 3.1 (PL-CE Training Aligns PL Rankings)**.** *Fix a query and its sampled reasoning trajectory set.* ***If the PL-CE objective is small, i.e., the learnable policy's set-conditional probabilities are close to the reward-shaped target on this set, then the PL ranking distributions induced by the policy and the target are close (uniformly over all permutations).*** *In particular, if the policy π_θ matches the optimal π^* on the set, then the two induced PL rankings are identical.*

Theorem 3.1 (formalized in Appendix G) controls the discrepancy between the induced PL rankings on a fixed pool through set-conditional alignment. The coefficient α allows us to modulate the optimization landscape: small α ensures robust coverage (minimizing the discrepancy bound), while larger α promotes mode seeking toward top candidates. Further explanation is in Appendix E.4.1.

3.2 Mass-aggregated Inference-time Scaling

Recall that our training objective aligns the model's within-set ranking over sampled reasoning trajectories. For inference, we need a matching decision rule to realize the corresponding objective in Eq. 1. We propose a simple and effective strategy with little computational overhead: we aggregate probability mass from trajectories to unique answers, enabling consistent selection with our set-conditional PL-CE training. Pseudo-code is presented in Algo. 2.

Mass-aggregated Ranking Scores. Given a test query $\boldsymbol{x}$, we generate N candidate trajectories $\mathcal{T}_N = \{\tau_1, \ldots, \tau_N\}$ with log probabilities $s_i := \log \pi_\theta(\tau_i \mid \boldsymbol{x})$. Let $g(\tau)$ map a reasoning trajectory to its final answer. For each distinct induced answer a, we define its *ranking score* by aggregating the mass of trajectories that produce a, controlled by a tunable $\lambda > 0$:

$$\text{Score}_\lambda(a) := \frac{\sum_{i:g(\tau_i)=a} \exp(\lambda s_i)}{\sum_{j=1}^{N} \exp(\lambda s_j)}. \qquad (8)$$

Finally, we output the top-K distinct answers under $\text{Score}_\lambda(a)$ as the final portfolio.

Table 1: Pass@3 and Pass@1 across 12 reasoning benchmarks, compared with diverse baselines. Each baseline samples $N = 10$ trajectories and randomly retains K trajectories; AUTOPORTFOLIO uses Mass Aggregation to retain up to K distinct answers from $N = 10$ trajectories for an end-to-end comparison of training and selection. RL baselines with our proposed mass aggregation scaling are shown in Fig. 6. The best and second-best results are shaded green (bold) and blue (underline) respectively; pretrained models are shown as context baselines.

Method / Pass@3 Performance	Mathematical Reasoning						Knowledge-Intensive Reasoning						Avg.	Rank
	AIME24	AIME25	MATH500	GSM8K	MATH	Omni-MATH	WebWalker	HQA	2Wiki.	MuSiQ.	Bamb.	GPQA-D		
Qwen2.5-72B	23.3	23.3	89.6	97.2	95.2	44.0	6.0	50.5	46.0	28.5	71.2	68.2	53.6	18
QwQ-32B	76.7	73.3	97.4	98.0	99.0	56.8	8.5	50.0	51.5	21.5	72.8	76.8	65.2	3
Qwen3-32B-thinking	83.3	73.3	96.8	97.4	99.2	56.6	9.0	42.0	50.5	23.0	66.4	78.3	64.7	5
DeepSeek-R1-32B	76.7	60.0	96.2	98.0	99.2	58.8	9.0	42.0	50.0	23.0	61.6	74.7	62.4	10
Qwen2.5-Math-7B	13.3	20.0	88.0	97.2	95.2	41.6	\	\	\	\	\	\	(59.2)	15
Deepseek-math-7b-rl	0.0	0.0	56.0	89.6	77.4	16.6	\	\	\	\	\	\	(39.9)	22
Search-r1	33.3	26.7	82.2	89.6	90.2	36.2	36.0	69.5	76.5	36.5	80.8	56.6	59.5	14
DeepResearcher	30.0	23.3	85.2	96.0	91.4	41.4	32.5	73.0	82.0	37.0	82.4	54.5	60.7	11
Tool-Light	36.7	30.0	87.0	95.8	92.6	36.2	47.0	72.0	81.5	40.5	85.6	63.1	64.0	7
Tool-Star	40.0	33.3	86.8	96.0	93.2	36.6	51.0	72.0	77.0	39.0	86.4	60.6	64.3	6
Llama3.1-8B-Instruct	10.0	3.3	63.0	90.2	77.6	19.2	30.5	62.5	71.5	31.5	82.4	55.6	49.8	21
+ GRPO	23.3	13.3	74.0	91.0	86.6	30.0	45.5	68.5	84.0	38.0	87.2	62.6	58.7	16
+ GRPO w/ Pass-k Training	16.7	13.3	64.0	91.0	80.6	22.6	38.0	67.5	79.5	35.5	79.2	54.5	53.5	19
+ *ARPO*	26.7	16.7	76.8	92.8	88.0	30.8	50.5	71.5	83.0	38.0	84.8	60.1	60.0	13
+ *AEPO*	10.0	13.3	69.0	94.6	85.2	28.6	50.0	71.5	82.0	41.5	84.8	69.7	58.4	17
+ AUTOPORTFOLIO	26.7	23.3	81.2	95.8	91.0	35.0	50.5	**75.0**	90.0	**49.5**	90.4	**77.3**	65.5	2
Qwen2.5-7B-Instruct	20.0	16.7	77.8	93.4	89.8	35.2	25.5	60.5	70.5	30.5	71.2	41.4	52.7	20
+ GRPO	30.0	33.3	85.0	96.0	92.2	38.2	48.5	68.5	79.0	39.0	83.2	61.1	62.8	9
+ GRPO w/ Pass-k Training	26.7	30.0	86.4	94.8	92.6	40.0	41.5	60.5	76.5	36.5	80.0	60.6	60.5	12
+ *ARPO*	36.7	**36.7**	82.6	93.4	90.0	36.8	40.0	71.5	82.0	37.5	81.6	69.2	63.2	8
+ *AEPO*	33.3	30.0	86.8	94.0	93.6	39.6	45.0	73.5	85.0	39.0	80.8	76.3	64.7	4
+ AUTOPORTFOLIO	**43.3**	**36.7**	**91.6**	**96.8**	**95.2**	**42.6**	**56.0**	74.5	**90.5**	45.0	**92.8**	69.7	**69.6**	**1**

Method / Pass@1 Performance	Mathematical Reasoning						Knowledge-Intensive Reasoning						Avg.	Rank
	AIME24	AIME25	MATH500	GSM8K	MATH	Omni-MATH	WebWalker	HQA	2Wiki.	MuSiQ.	Bamb.	GPQA-D		
Llama3.1-8B-Instruct	0.0	0.0	40.0	70.4	51.4	9.8	18.5	44.5	47.5	18.0	67.2	26.3	32.8	12
+ GRPO	3.3	10.0	58.8	85.8	74.6	22.0	36.0	66.5	77.0	30.5	79.2	38.9	48.6	7
+ GRPO w/ Pass-k Training	13.3	3.3	47.0	84.0	67.4	11.8	27.5	54.5	61.5	20.5	63.2	33.3	40.6	11
+ *ARPO*	23.3	10.0	60.2	85.2	78.4	19.4	34.5	65.5	73.0	25.0	80.8	32.3	49.0	6
+ *AEPO*	6.7	10.0	52.4	84.8	69.8	19.4	35.5	63.0	73.5	30.5	76.0	42.4	47.0	9
+ AUTOPORTFOLIO	16.7	23.3	72.6	89.8	84.0	23.2	**41.0**	64.5	**79.5**	**37.5**	77.6	46.0	54.6	2
Qwen2.5-7B-Instruct	13.3	3.3	67.2	85.6	81.8	25.2	17.5	51.0	54.0	20.0	60.0	30.3	42.4	10
+ GRPO	13.3	13.3	77.2	91.0	85.8	25.2	36.5	62.0	73.5	32.0	77.6	40.4	52.3	4
+ GRPO w/ Pass-k Training	13.3	13.3	73.4	92.4	84.4	25.0	28.5	54.0	63.5	25.5	64.8	37.9	48.0	8
+ *ARPO*	26.7	23.3	70.4	81.2	79.2	25.6	32.0	65.5	75.5	32.0	74.4	41.4	52.3	5
+ *AEPO*	16.7	23.3	76.8	91.4	88.6	26.0	33.5	59.5	71.5	29.5	74.4	**46.5**	53.1	3
+ AUTOPORTFOLIO	**30.0**	**33.3**	**82.0**	**94.4**	**89.0**	**31.4**	40.5	**68.0**	76.5	**37.5**	**83.2**	46.0	**59.3**	**1**

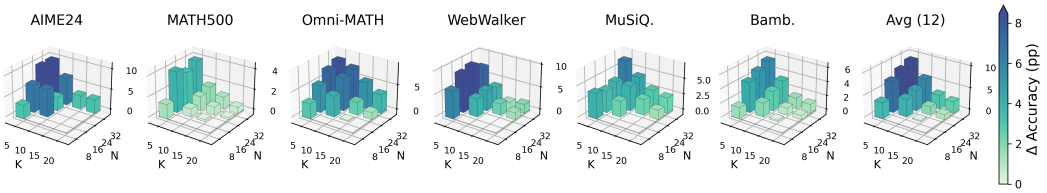

Figure 4: **Performance across budget N and number of answers K.** Each 3D bar plot reports the accuracy improvement relative to *random* under the corresponding (N, K) setting; invalid entries ($K > N$) are masked. The far-right panel summarizes the 12-benchmark average.

Training-consistent Answer Aggregation. Eq. 8 operates on the same finite candidate set optimized by PL-CE: PL-CE learns how to allocate probability mass across sampled trajectories, and inference aggregates that mass over trajectories leading to the same answer. When $\lambda = 1$, this recovers the answer-level marginal of the learned set-conditional trajectory distribution. λ *provides a lightweight test-time knob* without re-sampling: small λ (i.e., *broad support*) approaches vote-like aggregation by emphasizing broad support across trajectories; large λ (i.e., *peak confidence*) approaches max-probability selection by emphasizing the strongest trajectory within each answer cluster.

4 Experiments

We evaluate AUTOPORTFOLIO on three axes. (1) **Effectiveness**: Pass@1/Pass@3 comparisons with agentic RL baselines, tool-using agents, and pretrained models (Table 1; Tables 2,3). (2) **Scalability and Efficiency**: controlled sweeps over sampling budget N, returned answer count K, and token use test whether AUTOPORTFOLIO converts additional inference compute into better performance (Figs. 4,5,6,8). (3) **Mechanism**: λ sensitivity, selector comparisons, and validation dynamics test whether

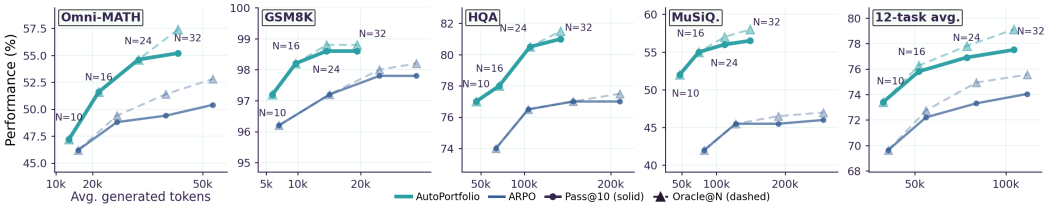

Figure 5: **Compute-normalized scaling.** Pass@10 and Oracle@N ($N \in \{10, 16, 24, 32\}$) are plotted against trajectory tokens under the same Mass Aggregation selection rule. Oracle@N is the accuracy when any of the N sampled candidates is correct, i.e., the method's *performance upper bound* at that budget N. AUTOPORTFOLIO scales more efficiently with budget and surpasses ARPO's oracle performance.

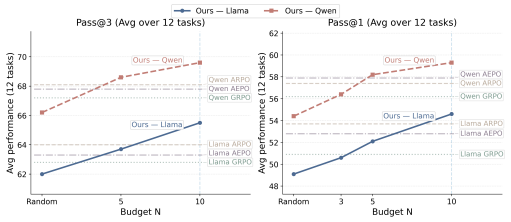

Figure 6: **Budget scaling on the 12-task average performance.** For Qwen2.5-7B, AUTOPORTFOLIO with $N = 5$ rollouts matches or exceeds baselines with $N = 10$ under the Mass Aggregation selection.

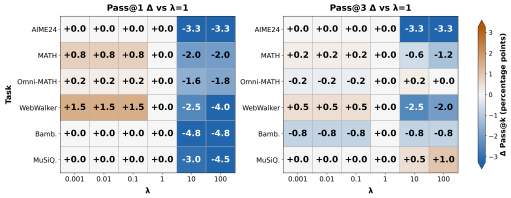

Figure 7: λ **sweep sensitivity.** Pass@1 and Pass@3 improvements relative to λ=1. $\lambda = 1$ is a stable default across benchmarks, balancing broad trajectory support and high probabilities.

gains come from within-set prioritization and scaling beyond generation alone (Appendix E.4.3; Figs. 7,9,11). We defer implementation details and additional experiments to Appendix E. Code will be made available at `https://github.com/yunzhe0306/AutoPortfolio`.

Datasets. We evaluate our method on 12 challenging benchmarks categorized into two domains:

- Mathematical Reasoning: *GSM8K* [12], *MATH* [23], *AIME 2024*, *AIME 2025*, *MATH-500* [31], and the more challenging *Omni-MATH* [19].
- Knowledge-Intensive Reasoning: *HotpotQA* (HQA) [69], *2WikiMultiHopQA* (2Wiki.) [24], *MuSiQue* (MuSiQ.) [57], and *Bamboogle* (Bamb.) [40]. We also include *GPQA-Diamond* (GPQA-D) [45] and *WebWalkerQA* [61] to evaluate search and expert-level question answering.

Baselines. We compare AUTOPORTFOLIO against a comprehensive set of baseline methods:

- Strong pre-trained models w/o tool usage: *Qwen2.5-72B* [67], *QwQ-32B* [56], *Qwen3-32B-thinking* [66], *DeepSeek-R1-32B* [22], *Qwen2.5-Math-7B* [68] and *Deepseek-math-7b-rl* [49].
- Reasoning agents w/ tools: *Search-r1* [27], DeepResearcher [83], *Tool-Light* [9], *Tool-Star* [16].
- Agentic RL methods: *GRPO* [49], *GRPO* with Pass@K training [55], *ARPO* [17], and *AEPO* [15]. We omit other RL baselines as prior agentic RL studies report that ARPO and AEPO are consistently stronger for agentic reasoning [17, 15].

4.1 Mathematics and Knowledge-Intensive Reasoning

AUTOPORTFOLIO consistently delivers strong performance against agentic RL baselines in Table 1 (also in Appendix Tables 2, 3). For an end-to-end comparison of PL-CE training with Mass-aggregated inference, AUTOPORTFOLIO uses scaling budget $N = 10$ in Table 1. We apply Pass@K to match budgeted sample-then-select evaluation, where any correct selected output counts as success. For baselines, we sample $N = 10$ candidate trajectories and draw K trajectories; RL baselines under Mass Aggregation scaling are shown in Fig. 6. The gains of AUTOPORTFOLIO persist across both backbones and both Pass@3/Pass@1, indicating that set-conditional alignment provides a reliable learning signal for inference-time scaling. On Qwen2.5-7B, AUTOPORTFOLIO ranks first overall in both metrics, improving the 12-task average over the strongest RL baseline from 64.7→69.6 for Pass@3 and 53.1→59.3 for Pass@1. On Llama3.1-8B, AUTOPORTFOLIO improves the 12-task average over the strongest same-backbone RL baseline from 60.0 to 65.5 for Pass@3 and from 49.0 to 54.6 for Pass@1. The improvements are pronounced in hard, knowledge-intensive settings where redundancy and correlated distractors make trajectory-level selection brittle. For example, on

Qwen2.5-7B, AUTOPORTFOLIO improves WebWalker over the strongest same-backbone RL baseline from 48.5 to 56.0 for Pass@3 and from 36.5 to 40.5 for Pass@1. The improvements also hold without inference-time scaling (Tables 2, 3). A case study of reasoning trajectories is in Appendix H. *Training Cost:* AUTOPORTFOLIO remains *comparable* to GRPO: for Qwen2.5-7B, maximum GPU memory is 334.7 vs. 333.3 GiB, average step time is 978.3s vs. 949.5s, and single-GPU peak memory is 90.1 vs. 91.3 GiB.

4.2 Analytical Experiments

Performance Dynamics across N and K. We vary both the candidate generation budget N and the returned answer count K in Fig. 4. Larger N generally amplifies gains by increasing the chance that the pool contains a correct answer, while improvements are strongest when a small set is selected from a larger pool, where ranking quality matters most. The largest absolute gains occur on complex benchmarks such as Omni-MATH, showing that AUTOPORTFOLIO is particularly effective in challenging regimes. With the same Mass Aggregation scaling, AUTOPORTFOLIO improves over ARPO under matched inference budgets (Fig. 8). Notably, AUTOPORTFOLIO can match or surpass ARPO Pass@32 even with much smaller selected answer sets (K) and fewer rollouts (N): its gains come from both better candidates generated and better use of the sampled candidate pool. ***Oracle-aware Scaling:*** Fig. 5 studies compute scaling as N varies. Under the same Mass Aggregation scaling, AUTOPORTFOLIO achieves stronger Pass@10 at comparable token budgets and can surpass ARPO's Oracle@N curve. Oracle@N is a method-specific coverage upper bound, separating candidate generation from ranking. AUTOPORTFOLIO can deliver larger token-normalized gains while staying close to its own oracle.

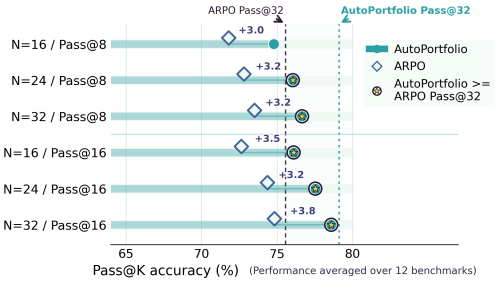

Figure 8: With the same Mass Aggregation scaling, AUTOPORTFOLIO consistently improves over ARPO across budgets. AUTOPORTFOLIO surpasses ARPO Pass@32 using a smaller budget N and lower selected-K settings, reducing the number of costly environment evaluations.

Characteristics Behind the Effectiveness of Mass Aggregation. Fig. 6 compares the performance of RL baselines and AUTOPORTFOLIO under the same Mass Aggregation scaling. AUTOPORTFOLIO improves consistently as the budget N increases. For Qwen2.5-7B, with only $N = 5$ rollouts, it matches or exceeds GRPO/ARPO/AEPO with $N = 10$, demonstrating better budget efficiency while retaining most gains from larger-budget inference. Fig. 7 studies the aggregation parameter λ, which interpolates between mode-covering (small λ) and mode-seeking (large λ). $\lambda = 1$ is an overall stable default aligned with PL-CE training; very large λ can reintroduce trajectory-level brittleness, while overly small λ can miss correct answers supported by a few high-quality trajectories. Under a fixed sampling budget across both backbones, Mass Aggregation can consistently outperform trajectory top-log-probability and an *external Qwen3-8B verifier*, without additional verifier computation (Appendix E.4.3).

Validation Rewards. We also track validation reward dynamics (smoothed) during training in Fig. 9. AUTOPORTFOLIO can improve faster and reach higher validation reward, indicating that PL-CE provides a more effective signal for within-set prioritization. The larger gap on Best@5 than Best@2 further supports that set-conditional alignment matters most when the candidate pool becomes more crowded and correlated distractors are more likely to appear.

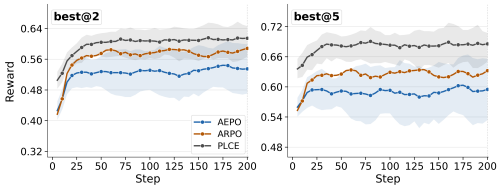

Figure 9: Validation reward during 200-step training.

5 Related Works

Agentic Reasoning via RL Training. Recent efforts have significantly enhanced language model capabilities through RL-based alignment [35, 86] and reasoning traces that enable supervision signals [60, 8]. Building on PPO-style updates [47], reasoning-focused variants such as GRPO [49] and subsequent alternatives [72, 32] have shown strong gains on mathematical and coding-style reasoning, while GiGPO extends group-based updates to long-horizon agent training [18]. Despite these advances, the dominant objective class still maximizes the expected return of individual sampled

trajectories, which can encourage mode-seeking and reduced diversity under sharp or noisy reward landscapes [59, 65]. Distribution-matching alternatives mitigate over-optimization by matching reward distributions, yet still do not directly optimize *set-wise* ranking and prioritization [85, 54]. Recent agentic optimizers further improve stability for multi-step tool-using agents, but remain primarily trajectory-wise, rather than explicitly optimizing set-wise ranking over candidates [15, 17]. In contrast, our AUTOPORTFOLIO explicitly optimizes the *joint quality* of a candidate set via set-wise ranking, enabling efficient and effective scaling at inference time.

Inference-time Scaling for Reasoning Agents. Trading test-time computation for performance ranges from sampling strategies, such as Self-Consistency [58] to search-based decoding and planning methods [70, 1, 13, 71, 10, 75]. However, the efficacy of large test-time budgets is often bottlenecked by *selection*, since simple heuristics like max log-probability can plateau as N grows [4, 74]. Some works also interleave generation with explicit tree-search or planning, guided by reward models, discriminators, environment feedback, or self-reflection, improving both depth and reliability at increased test-time compute [84, 26, 42, 29]. These approaches tightly couple generation and selection via external orchestration. In contrast, our framework internalizes this coupling *during training*, so the agent natively generates and ranks a high-quality reasoning portfolio without relying on heavy external inference-time mechanisms. A complementary discussion is in Appendix D.

6 Conclusion

We introduced AUTOPORTFOLIO, a framework that shifts agentic policy learning from trajectory-wise optimization to set-conditional portfolio alignment. By internalizing Plackett-Luce ranking logic into the training objective, AUTOPORTFOLIO balances mode-covering diversity with mode-seeking precision, prioritizing high-quality reasoning paths within each sampled set. Empirical evaluations across 12 challenging reasoning benchmarks show that AUTOPORTFOLIO consistently outperforms strong agentic RL baselines, especially when paired with our lightweight Mass-aggregated inference scaling strategy. Our findings suggest that aligning training objectives with set-conditional selection dynamics offers a theoretically grounded and scalable path toward more reliable agentic reasoning.

Acknowledgments

This work is supported by National Science Foundation under Award No. IIS-2117902. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

References

[1] Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Wang. Swe-search: Enhancing software agents with monte carlo tree search and iterative refinement. *arXiv preprint arXiv:2410.20285*, 2024.

[2] Wenxuan Bao, Yanjun Zhao, Xiyuan Yang, and Jingrui He. Ramen: Robust test-time adaptation of vision-language models with active sample selection, 2026.

[3] Edoardo Botta, Yuchen Li, Aashay Mehta, Jordan T Ash, Cyril Zhang, and Andrej Risteski. On the query complexity of verifier-assisted language generation. *arXiv preprint arXiv:2502.12123*, 2025.

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

[5] Shreyas Chaudhari, Pranjal Aggarwal, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, Karthik Narasimhan, Ameet Deshpande, and Bruno Castro da Silva. Rlhf deciphered: A critical analysis of reinforcement learning from human feedback for llms. *ACM Computing Surveys*, 58(2):1–37, 2025.

[6] Huayu Chen, Guande He, Lifan Yuan, Ganqu Cui, Hang Su, and Jun Zhu. Noise contrastive alignment of language models with explicit rewards. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[7] Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International conference on machine learning*, pages 1042–1051. PMLR, 2019.

[8] Sirui Chen, Yunzhe Qi, Haobo Xu, Jiaru Zou, Mengting Ai, Zihao Li, and Jingrui He. Proposing better rollouts for on-policy distillation. *arXiv preprint arXiv:2610.36477*, 2026.

[9] Yifei Chen, Guanting Dong, and Zhicheng Dou. Toward effective tool-integrated reasoning via self-evolved preference learning. *arXiv preprint arXiv:2509.23285*, 2025.

[10] Zhiliang Chen, Xinyuan Niu, Chuan-Sheng Foo, and Bryan Kian Hsiang Low. Broaden your scope! efficient multi-turn conversation planning for llms with semantic space. *arXiv preprint arXiv:2503.11586*, 2025.

[11] Ziru Chen, Michael White, Ray Mooney, Ali Payani, Yu Su, and Huan Sun. When is tree search useful for llm planning? it depends on the discriminator. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13659–13678, 2024.

[12] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[13] Xing Cui, Yueying Zou, Zekun Li, Peipei Li, Xinyuan Xu, Xuannan Liu, and Huaibo Huang. Tˆ 2agent a tool-augmented multimodal misinformation detection agent with monte carlo tree search. *arXiv preprint arXiv:2505.19768*, 2025.

[14] Chris Dann, Yishay Mansour, Mehryar Mohri, Ayush Sekhari, and Karthik Sridharan. Guarantees for epsilon-greedy reinforcement learning with function approximation. In *International conference on machine learning*, pages 4666–4689. PMLR, 2022.

[15] Guanting Dong, Licheng Bao, Zhongyuan Wang, Kangzhi Zhao, Xiaoxi Li, Jiajie Jin, Jinghan Yang, Hangyu Mao, Fuzheng Zhang, Kun Gai, et al. Agentic entropy-balanced policy optimization. *arXiv preprint arXiv:2510.14545*, 2025.

[16] Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *arXiv preprint arXiv:2505.16410*, 2025.

[17] Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026.

[18] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[19] Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[20] Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*, 2025.

[21] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[22] 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*, 2025.

[23] 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.

[24] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*, 2020.

[25] Kihyuk Hong, Yuhang Li, and Ambuj Tewari. A primal-dual-critic algorithm for offline constrained reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 280–288. PMLR, 2024.

[26] Chia-Yu Hung, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. Inference time alignment with reward-guided tree search. *arXiv preprint arXiv:2406.15193*, 2024.

[27] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

[28] Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, Peifeng Wang, Silvio Savarese, et al. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *arXiv preprint arXiv:2504.09037*, 2025.

[29] Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*, 2024.

[30] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, RÃl'mi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[31] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

[32] Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, et al. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026.

[33] R Duncan Luce et al. *Individual choice behavior*, volume 4. Wiley New York, 1959.

[34] Ning Miao, Yee Whye Teh, and Tom Rainforth. Selfcheck: Using LLMs to zero-shot check their own step-by-step reasoning. In *The Twelfth International Conference on Learning Representations*, 2024.

[35] 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.

[36] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750, 2007.

[37] Aske Plaat, Max van Duijn, Niki Van Stein, Mike Preuss, Peter van der Putten, and Kees Joost Batenburg. Agentic large language models, a survey. *Journal of Artificial Intelligence Research*, 84, 2025.

[38] Robin L Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.

[39] Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O Arik. CHASE-SQL: Multi-path reasoning and preference optimized candidate selection in text-to-SQL. In *The Thirteenth International Conference on Learning Representations*, 2025.

[40] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, 2023.

[41] Yunzhe Qi, Sirui Chen, Jiaru Zou, and Jingrui He. Auto-regressive in-context demonstration selection, 2026.

[42] Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv:2408.06195*, 2024.

[43] Yun Qu, Qi Wang, Yixiu Mao, Heming Zou, Yuhang Jiang, Yingyue Li, Wutong Xu, Lizhou Cai, Weijie Liu, Clive Bai, et al. Listwise policy optimization: Group-based rlvr as target-projection on the llm response simplex. *arXiv preprint arXiv:2605.06139*, 2026.

[44] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

[45] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[46] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

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

[48] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

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

[50] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[51] Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

[52] Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. Fast best-of-n decoding via speculative rejection. *Advances in Neural Information Processing Systems*, 37:32630–32652, 2024.

[53] Kyle Swanson, Gary Liu, Denise B Catacutan, Autumn Arnold, James Zou, and Jonathan M Stokes. Generative ai for designing and validating easily synthesizable and structurally novel antibiotics. *Nature machine intelligence*, 6(3):338–353, 2024.

[54] 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. *arXiv preprint arXiv:2602.02710*, 2026.

[55] Yunhao Tang, Kunhao Zheng, Gabriel Synnaeve, and Remi Munos. Optimizing language models for inference time objectives using reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

[56] Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown. *Hugging Face*, 2024.

[57] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. âŹń musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

[58] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[59] Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. Robust imitation of diverse behaviors. *Advances in Neural Information Processing Systems*, 30, 2017.

[60] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[61] Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, et al. Webwalker: Benchmarking llms in web traversal. *arXiv preprint arXiv:2501.07572*, 2025.

[62] Kehan Wu, Yingce Xia, Pan Deng, Renhe Liu, Yuan Zhang, Han Guo, Yumeng Cui, Qizhi Pei, Lijun Wu, Shufang Xie, et al. Tamgen: drug design with target-aware molecule generation through a chemical language model. *Nature Communications*, 15(1):9360, 2024.

[63] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

[64] Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying llm-based software engineering agents. *Proceedings of the ACM on Software Engineering*, 2(FSE):801–824, 2025.

[65] Wanqi Xue, Bo An, Shuicheng Yan, and Zhongwen Xu. Reinforcement learning from diverse human preferences. *arXiv preprint arXiv:2301.11774*, 2023.

[66] 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.

[67] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[68] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

[69] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380, 2018.

[70] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[71] Jaesik Yoon, Hyeonseo Cho, Doojin Baek, Yoshua Bengio, and Sungjin Ahn. Monte carlo tree diffusion for system 2 planning. *arXiv preprint arXiv:2502.07202*, 2025.

[72] 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. *arXiv preprint arXiv:2503.14476*, 2025.

[73] Longfei Yun, Chenyang An, Zilong Wang, Letian Peng, and Jingbo Shang. The price of format: Diversity collapse in llms. *arXiv preprint arXiv:2505.18949*, 2025.

[74] Jiaxin Zhang, Caiming Xiong, and Chien-Sheng Wu. Agentic confidence calibration. *arXiv preprint arXiv:2601.15778*, 2026.

[75] Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23378–23386, 2025.

[76] Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, et al. Genprm: Scaling test-time compute of process reward models via generative reasoning. *arXiv preprint arXiv:2504.00891*, 2025.

[77] Kai Zhao, Yanjun Zhao, Jiaming Song, Shien He, Lusheng Zhang, Qiang Zhang, and Tianjiao Li. Saber: Switchable and balanced training for efficient llm reasoning, 2025.

[78] Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor W. Tsang. Second-order fine-tuning without pain for llms:a hessian informed zeroth-order optimizer, 2025.

[79] Yanjun Zhao, Ruizhong Qiu, Tianxin Wei, Yuanchen Bei, Zhining Liu, Lingjie Chen, Ismini Lourentzou, Hanghang Tong, and Jingrui He. Recontext: Recursive evidence replay as llm harness for long-context reasoning, 2026.

[80] Yanjun Zhao, Tianxin Wei, Jiaru Zou, Xuying Ning, Yuanchen Bei, Lingjie Chen, Simmi Rana, Wendy H. Yang, Hanghang Tong, and Jingrui He. Papermind: Benchmarking agentic reasoning and critique over scientific papers in multimodal llms, 2026.

[81] Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

[82] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*, 2024.

[83] Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 414–431, 2025.

[84] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

[85] Xuekai Zhu, Daixuan Cheng, Dinghuai Zhang, Hengli Li, Kaiyan Zhang, Che Jiang, Youbang Sun, Ermo Hua, Yuxin Zuo, Xingtai Lv, et al. Flowrl: Matching reward distributions for llm reasoning. *arXiv preprint arXiv:2509.15207*, 2025.

[86] Jiaru Zou, Yunzhe Qi, Sirui Chen, Federico Bianchi, Shang Zhu, Pan Lu, Jingrui He, and James Zou. Hybrid reasoning agents. *arXiv preprint arXiv:2610.18861*, 2026.

[87] Jiaru Zou, Ling Yang, Yunzhe Qi, Sirui Chen, Mengting Ai, Ke Shen, Jingrui He, and Mengdi Wang. Autotool: Dynamic tool selection and integration for agentic reasoning. *arXiv preprint arXiv:2512.13278*, 2025.

A Pseudo Code

Algorithm 1 AUTOPORTFOLIO Training

Inputs: Training Dataset (queries & answers). Policy π_θ, Reference Policy π_{ref}.
Parameters: Training Rollout Size $|\mathcal{T}|$, KL Coefficient β, Balance Coefficient α.
for each training iteration **do**
  Initialize a temporary rollout buffer.
  ▷ **Collecting Trajectories & Evaluation**
  for each query $\boldsymbol{x}$ in a *rollout batch* **do**
    Sample $\mathcal{T} = \{\tau_1, \ldots, \tau_{|\mathcal{T}|}\} \sim \pi_\theta(\cdot \mid \boldsymbol{x})$.
    Compute rewards $r_i \leftarrow r(\boldsymbol{x}, \tau_i),\ \forall \tau_i \in \mathcal{T}$.
    Store $(\boldsymbol{x}, \mathcal{T}, \{r_i\})$ into the rollout buffer.
  end for
  ▷ **Set-Conditional Alignment (PL-CE)**
  for each sampled batch $\mathcal{D}$ of training tuples from the rollout buffer **do**
    Initialize batch losses: $\mathcal{L}_{\text{fwd}} \leftarrow 0$, $\mathcal{L}_{\text{rev}} \leftarrow 0$.
    for tuple $(\boldsymbol{x}, \mathcal{T}, \{r_i\}) \in \mathcal{D}$ **do**
      (1) Compute Target Distribution q on $\mathcal{T}$:
        $u_i \leftarrow \log \pi_{\text{ref}}(\tau_i \mid \boldsymbol{x}) + r_i/\beta,\ \forall \tau_i \in \mathcal{T}$.
        $q_i \leftarrow \exp(u_i) / \sum_{\tau_j \in \mathcal{T}} \exp(u_j)$.
      (2) Compute Policy Distribution p on $\mathcal{T}$:
        $s_i \leftarrow \log \pi_\theta(\tau_i \mid \boldsymbol{x}),\ \forall \tau_i \in \mathcal{T}$.
        $p_i \leftarrow \exp(s_i) / \sum_{\tau_j \in \mathcal{T}} \exp(s_j)$.
      (3) Accumulate Forward & Reverse CE (Eq. 7):
        $\mathcal{L}_{\text{fwd}} \leftarrow \mathcal{L}_{\text{fwd}} - \sum_{\tau_i \in \mathcal{T}} q_i \log p_i$.
        $\mathcal{L}_{\text{rev}} \leftarrow \mathcal{L}_{\text{rev}} - \sum_{\tau_i \in \mathcal{T}} p_i \log q_i$.
    end for
    Normalize over batch: $\mathcal{L}_{\text{fwd}} \leftarrow \mathcal{L}_{\text{fwd}}/|\mathcal{D}|,\ \mathcal{L}_{\text{rev}} \leftarrow \mathcal{L}_{\text{rev}}/|\mathcal{D}|$.
    Total loss: $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{fwd}} + \alpha \cdot \mathcal{L}_{\text{rev}}$.
    Update policy parameters θ with $\mathcal{L}_{\text{total}}$.
  end for
end for

Algorithm 2 AUTOPORTFOLIO Inference-Time Scaling (Portfolio Generation)

Inputs: Test Query $\boldsymbol{x}$, Policy π_θ.
Parameters: Samples N, Returned answer count K, Aggregation coefficient λ.
▷ **Candidate Generation**
Sample candidates $\mathcal{T}_N = \{\tau_1, \ldots, \tau_N\} \sim \pi_\theta(\cdot|\boldsymbol{x})$.
Retrieve $s_i \leftarrow \log \pi_\theta(\tau_i \mid \boldsymbol{x}),\ \forall i = 1, \ldots, N$.
▷ **Answer-mass Aggregation**
Let $g(\tau)$ map a trajectory to its final answer.
Extract unique answers $\mathcal{U} \leftarrow \text{Unique}(\{g(\tau_i) | \tau_i \in \mathcal{T}_N\})$.
for answer $a \in \mathcal{U}$ **do**
  Compute ranking score (Eq. 8):
    $\text{Score}_\lambda(a) \leftarrow \frac{\sum_{i:g(\tau_i)=a} \exp(\lambda s_i)}{\sum_{j=1}^{N} \exp(\lambda s_j)}$.
end for
▷ **Portfolio Generation w/ Top-K Distinct Answer(s)**
return the K answers $a \in \mathcal{U}$ with largest $\text{Score}_\lambda(a)$.

B Declaration of LLM Usage and Broader Impact

Declaration of LLM Usage. This work studies LLM agents, where LLMs are central experimental subjects throughout the paper: they are used as policy backbones, trained with our proposed objective and baseline RL methods, sampled to generate reasoning trajectories, and evaluated on mathematical and knowledge-intensive reasoning benchmarks. LLMs were also used as editing tools to improve manuscript clarity and grammar correctness.

Broader Impact. This paper introduces AUTOPORTFOLIO, a set-conditional policy alignment framework for improving sample-then-select inference-time scaling in reasoning agents. By helping

models better prioritize and aggregate candidate reasoning trajectories, our method can improve the reliability and budget efficiency of LLM-based reasoning systems in domains such as mathematics and knowledge-intensive question answering. While more capable reasoning agents may inherit general risks associated with LLM deployment, such as misuse or over-reliance on automated outputs, our work does not introduce new high-risk application settings or additional societal concerns beyond those of existing reasoning-agent systems.

C Future Directions and Limitations

Future Directions. A natural next step is adaptive portfolio construction, where the agent chooses N, K, and aggregation sharpness per query under token or latency constraints, connecting AUTOPORTFOLIO to compute-optimal test-time scaling [50, 63]. Another direction is to combine PL-CE with richer supervision, including process-level verifiers or generative process reward models, so that set-conditional targets can use both final correctness and intermediate reasoning quality [31, 48, 76]. AUTOPORTFOLIO is also complementary to explicit search and planning: portfolio alignment could prioritize candidates produced by tree-style exploration while preserving the lightweight selection objective [70, 11].

Limitations. AUTOPORTFOLIO is designed for sample-then-select inference, where a finite candidate set is generated first and selection is then performed within that set to achieve effective and efficient inference-time scaling. This scope matches the deployment regime studied in this paper, but it does not naturally cover all kinds of inference-time scaling paradigms out of the box, such as tightly coupled tree search or multi-stage planning. Our experiments also focus on mathematical and knowledge-intensive reasoning, where rewards and answer equivalence can be defined reliably. PL-CE can still train on reward-labeled trajectories when answer equivalence is ambiguous, but Mass Aggregation requires a reliable grouping rule. Otherwise, selection can operate on trajectories instead. Coding tasks could use execution- or test-based equivalence; long-form outputs may require semantic or verifier-defined clustering, whose additional errors and costs are not evaluated in this work. Meanwhile, our experiments do not include multiple independent training seeds because of high computational costs and search-API costs for RL training. We leave these extensions as important and promising future directions of this paper.

D Related Works (Continued)

Agentic Reasoning via Reinforcement Learning. Recent RL efforts have provided a practical foundation for improving language model reasoning [35, 77, 78], while explicit reasoning traces (e.g., Chain-of-Thought traces) further unlocked verifiable supervision signals for downstream optimization. Most modern agentic-RL systems build on PPO-style trust-region updates [47], but adapt them to the long-CoT and long-horizon regime: GRPO replaces critic-based advantage estimation with group-relative comparisons, improving memory efficiency and yielding strong gains in mathematical reasoning [49]; DAPO further targets the instability of large-scale long-CoT RL with decoupled clipping and dynamic sampling to prevent entropy collapse and stabilize optimization [72]. Moving from single-turn reasoning to multi-step agents, GiGPO introduces a hierarchical "group-in-group" structure that combines episode-level relative advantages with step-level grouping over repeated states, enabling finer-grained credit assignment while retaining critic-free, group-based updates [18]. When multiple reward dimensions are optimized jointly, GDPO argues that directly applying group normalization can collapse distinct reward combinations into indistinguishable advantages, and proposes reward-decoupled normalization to preserve signal resolution and improve stability [32]. One concurrent work, LPO [43], also formulates group-based learning as target projection on a finite response simplex. It uses policy-to-behavior likelihood-ratio coordinates and separates target construction from the projection divergence.

Despite these advances, the dominant objective class still optimizes the expected return of a single sampled trajectory (or stepwise surrogates), which can implicitly encourage mode-seeking and reduced diversity under diverse preference settings [59, 65]; distributional alternatives such as FlowRL target the reward distribution to mitigate mode collapse, yet still do not directly and explicitly optimize intra-set prioritization over a candidate portfolio [85]. Recent agentic optimizers improve the stability and exploration dynamics of multi-step tool-using agents, but remain primarily trajectory-wise and do not explicitly optimize *set-wise* ranking over the candidate set (e.g., AEPO [15] and

ARPO [17]). In contrast, our AUTOPORTFOLIO explicitly optimizes the joint quality of a candidate set via set-wise ranking, directly reallocating probability mass on the simplex across candidates, which enables intrinsic selection under inference-time scaling.

Inference-time Scaling for Reasoning Agents. Trading test-time compute for performance has become an effective paradigm, ranging from repeated sampling to voting or selection (e.g., self-consistency) [58, 2], to explicit search-based decoding and planning, including Tree-of-Thought [70] and MCTS-style exploration in tool-using [1, 13, 71, 10, 75, 79, 80]. While larger test-time budgets can substantially improve success rates, recent evidence suggests that these gains are often bottlenecked by the selection mechanism, i.e., how we identify the best trajectory or solution among many candidates, especially when simple heuristics (e.g., maximum log-probability) can saturate at larger N [4, 74].

This has motivated two complementary directions. First, multiple analyses now view inference compute as an independent perspective and study compute-optimal ways of allocating sampling or search effort across prompts, demonstrating strong improvements from repeated sampling, while also highlighting diminishing returns without stronger ranking signals [4, 63, 50]. Second, a rapidly growing line strengthens selection via learned verifiers and process-level reward models (PRMs), where stepwise feedback guides rejection, refinement, and search rather than only selecting among completed answers [48, 76, 3]. Related works also improve the *efficiency* of test-time selection (e.g., accelerating Best-of-N-style decoding via speculative rejection), tightening the accuracy-compute trade-off [52]. In parallel, agentic inference-time scaling methods interleave generation with explicit tree-search or planning, guided by learned signals or self-reflection, improving both depth and reliability at increased test-time compute [84, 26, 42, 29]. Crucially, these approaches typically couple generation and selection via external orchestration (e.g., maintaining a search tree, invoking evaluators and verifiers, or running MCTS loops), and recent agentic tree-search evidence further emphasizes that performance depends critically on verifier quality [84, 11]. In contrast, our framework internalizes this coupling during training, so the agent natively generates and ranks a high-quality reasoning portfolio without relying on heavy external test-time mechanisms.

E Experiments (Continued)

E.1 Datasets

Mathematical Reasoning:

- **GSM8K** [12] is a grade-school math word-problem benchmark that tests multi-step arithmetic reasoning with short, exact final answers. We use it as a standard sanity check for whether a policy can reliably reach correct numeric conclusions under limited context.
- **MATH** [23] contains competition-style problems spanning diverse topics (e.g., algebra, geometry, number theory) and difficulty levels. It stresses longer-horizon symbolic reasoning and is a widely used reference point for frontier math progress.
- **AIME 2024** (American Invitational Mathematics Examination (AIME) contest problems) provides olympiad-style, short-answer math questions that are often used as a harder, out-of-distribution probe than standard training-style benchmarks. We treat it as a frontier evaluation of reliability under high difficulty and limited redundancy in the question format.
- **AIME 2025** (American Invitational Mathematics Examination (AIME) contest problems) similarly evaluates very challenging short-answer math reasoning, with problem distributions distinct from typical web-math corpora. Including both 2024 and 2025 helps reduce single-year idiosyncrasies and better reflect frontier-level generalization.
- **MATH-500** [31] is a curated subset of 500 MATH problems that is commonly used for efficient, reproducible evaluation. It offers a stable testbed for controlled ablations while remaining meaningfully challenging.
- **Omni-MATH** [19] targets broader and more difficult math reasoning, emphasizing coverage across topics and stronger generalization demands. We include it to probe whether improvements persist beyond standard math benchmarks.

Knowledge-intensive Reasoning:

- **HotpotQA** [69] evaluates multi-hop, knowledge-intensive question answering that typically requires combining evidence across multiple documents. It directly tests whether an agent can maintain coherent reasoning over retrieved facts.
- **2WikiMultiHopQA** [24] is a Wikipedia-based multi-hop QA benchmark designed to require reasoning across two linked entities/pages. It complements HotpotQA by providing a different construction pipeline and hop structure.
- **MuSiQue** [57] constructs multi-hop questions by composing reasoning questions, explicitly controlling for multi-step reasoning requirements. This reduces shortcut artifacts and makes within-set ranking failures more visible under sampling.
- **Bamboogle** [40] focuses on compositional and multi-step factual reasoning where models can be confidently wrong in systematic ways. It is useful for testing correlated failure patterns under repeated sampling.
- **GPQA-Diamond** [45] is an expert-level QA benchmark, reducing the chance that shallow retrieval or memorization suffices. We use it to evaluate deep knowledge reasoning under high difficulty.
- **WebWalkerQA** [61] evaluates web navigation and deep search reasoning where success depends on finding and synthesizing evidence from the open web. It serves as a tool-intensive counterpart to static multi-hop QA benchmarks.

E.2 Baselines

- **Qwen2.5-72B** [67] is a strong open-weights reasoning-capable model that provides a high-capacity baseline for pure generation quality. We use it to benchmark performance without additional training in our framework.
- **QwQ-32B** [56] is a reasoning-oriented model variant intended to produce stronger multi-step solutions than general instruction-tuned counterparts at similar scale. It serves as a competitive "reasoning-first" reference point for inference-time sampling.
- **Qwen3-32B-thinking** [66] represents a "thinking" variant optimized for longer-horizon reasoning traces at inference. We include it to compare against modern and latest reasoning-specialized checkpoints.
- **DeepSeek-R1-32B** [22] is a frontier reasoning baseline known for strong chain-of-thought style problem solving. It anchors comparisons to the current generation of reasoning-focused model families.
- **Qwen2.5-Math-7B** [68] is a domain-specific model optimized for advanced mathematical problem-solving. It serves as a rigorous baseline for assessing the impact of focused mathematical instruction tuning, relative to general reasoning capabilities at the 7B scale.
- **Deepseek-math-7b-rl** [49] is a reinforcement learning-enhanced model designed to improve mathematical reasoning through GRPO. It acts as a key reference point for evaluating RL-based alignment strategies in quantitative domains.
- **Search-r1** [27] is a pretrained agent that explicitly uses web search to answer questions. We include it as a representative tool-using reasoning agent where performance is shaped by both generation and retrieval decisions.
- **DeepResearcher** [83] is an end-to-end RL-trained research agent that learns to search, browse, and synthesize evidence directly in dynamic open-web environments. It provides a strong reference for training-based tool-augmented reasoning beyond prompt-engineered or static-RAG search agents.
- **Tool-Light** [9] studies lightweight tool-integrated reasoning with training mechanisms aimed at improving tool use without heavy inference-time search. We include it to benchmark agentic performance with relatively efficient tool interaction.
- **Tool-Star** [16] is a tool-augmented agent baseline emphasizing improved tool invocation and reasoning reliability. It helps contextualize our gains against strong tool-using agent pipelines.
- **Qwen2.5-7B-Instruct** [67] is an open-weights instruction-tuned base model used as a practical policy initialization at moderate scale. It lets us test whether our training objective improves within-set ranking beyond what the model already provides.
- **Llama-3.1-8B-Instruct** [21] is a widely used instruction-tuned baseline with strong general capabilities at small scale. Using it as an initialization checks that results are not specific to one model family.
- **GRPO** [49] is a reasoning-focused RL fine-tuning baseline that normalizes rewards within sampled groups to reduce variance and improve stability. We treat it as a strong representative of modern agentic RL for math and reasoning.

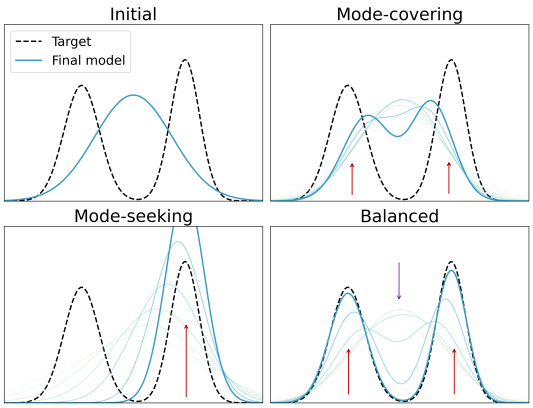

Figure 10: **Mode-covering vs. mode-seeking updates (concept illustration).** Dashed curves denote a reward-shaped *target* distribution with two modes, while solid curves show the resulting policy distribution after different training emphases. Starting from an *initial* policy that misses the bimodal structure, a *mode-covering* update spreads probability mass to cover both high-quality regions (often at the cost of over-smoothing the valley), whereas a *mode-seeking* update concentrates mass on a single dominant mode and can ignore the other. A *balanced* update reallocates mass to raise both target modes while suppressing low-quality regions, yielding a distribution that better matches the target shape.

- **GRPO with Pass@K Training** [55] explicitly trains the policy to optimize inference-time k-sample objectives (including Pass@k and majority voting) by treating the downstream aggregation rule as the training objective, rather than only optimizing expected reward per trajectory.
- **ARPO** [17] is a recent agentic policy optimization variant designed to improve stability/capability for multi-step agents under RL fine-tuning. It provides a modern point of comparison among recent reasoning-optimizer families.
- **AEPO** [15] is a recent agentic optimizer that explicitly balances exploration/entropy and learning signals for agent training. We include it to compare against state-of-the-art agentic RL variants beyond conventional GRPO-style group normalization.

E.3 Complementary Implementation Details of Experiments

Cold-start Supervised Fine-tuning. Following existing work [51], we initialize our policy π_θ from different base models and then apply cold-start SFT to enable tool utilization. We apply the cold-start training dataset from existing works [17, 15]. We train our base models using the LlamaFactory [82] with a learning rate of 7×10^{-6}. The batch size is set to 128, with a weight decay of 0.1, and the model is trained in SFT style for 3 epochs. We apply a maximum input length of 4096 tokens.

RL Training Details. We follow a similar protocol and environment setting as [17, 15]. For our framework as well as all the RL baselines, we run on NVIDIA GH200 GPUs, using a training batch size of 64. Prompts are capped at 1536 tokens and responses at 4096 tokens. During training trajectory generation, we sample 16 trajectories per query with temperature $T = 1.0$ for all the RL methods. Our RL training dataset is the 10K open-source dataset from [16]. Initial learning rate is fixed to 1×10^{-6} across different methods, which is also the default setup in [17, 15].

For the Set-Conditional Alignment, we compute the PL-CE loss (Eq. 7) with a KL coefficient $\beta = 1$. The balance coefficient α is set to 0.05 across different experiments. At inference time, we apply our Mass-aggregated selection $\mathrm{Score}_\lambda(a) \propto \sum_{i:g(\tau_i)=a} \exp\big(\lambda\, s_i/\omega_{\mathrm{median}}\big)$ with $\lambda = 1.0$, where ω is a dataset-level scale computed as the median (over evaluation queries) of the per-query log-probability range $\max_i s_i - \min_i s_i$ across sampled trajectories. Models are trained for 200 steps on a curated mixture of open-source mathematical and reasoning datasets. We adopt the same reward derivation procedure as in [17]. We consider the Python coding tool and the real-time search tool. Here, we use the Google Search API to retrieve real-time search results, and the API reports the top-10 result snippets in JSON format. We are dedicated to releasing our implementation and source code after paper publication.

E.4 Complementary Experiment Results

E.4.1 Concept Illustration of Mode Seeking and Covering

Here, Fig. 10 highlights the central tension that arises when learning from a sampled trajectory pool: the desired set-conditional target can be multi-modal, but different optimization pressures can move probability mass in qualitatively different ways. In the *mode-covering* regime, the update encourages placing non-trivial mass wherever the target assigns mass, which helps ensure that multiple high-quality modes remain represented in the policy. In the *mode-seeking* regime, the update penalizes probability assigned to relatively low-target regions more aggressively, which sharpens the distribution but risks collapsing onto a single attractive mode and discarding alternative correct strategies. A *balanced* objective mediates these behaviors by explicitly reallocating mass across candidates on the simplex: increasing mass on under-allocated, reward-aligned modes necessarily comes with decreasing mass on over-allocated (or low-quality) regions, keeping the total normalized mass intact. As a result, the policy can better match the target's multi-modal structure, which is precisely the behavior needed when inference-time scaling depends on reliable *within-set* prioritization among many competing trajectories.

E.4.2 Math and Knowledge-Intensive Reasoning (Continued)

In Tables 2 and 3, we further include the complete experiment results for Pass@1 and Pass@3. Here, AUTOPORTFOLIO is given different scaling budgets: "Random" means no inference-time scaling, and $N = 3, 5, 10$ denote scaling budgets of $3, 5, 10$, respectively. For other baselines, we randomly draw $N = 10$ candidate trajectories (and their corresponding answers) and then sample 1 or 3 answers to calculate the performance. The results show that AUTOPORTFOLIO can generally deliver the best overall results compared with baselines.

Table 2: Comparison with agentic RL methods. Overall Pass@1 and Pass@3 results on 12 challenging reasoning tasks are presented. The top two outcomes are bolded and underlined.

Method / Pass@3 Performance	**Mathematical Reasoning**						**Knowledge-Intensive Reasoning**						**Avg.**
	AIME24	**AIME25**	**MATH500**	**GSM8K**	**MATH**	**Omni-MATH**	**WebWalker**	**HQA**	**2Wiki.**	**MuSiQ.**	**Bamb.**	**GPQA-D**	
Llama3.1-8B-Instruct	10.0	3.3	63.0	90.2	77.6	19.2	30.5	62.5	71.5	31.5	82.4	55.6	49.8
+ GRPO	23.3	13.3	74.0	91.0	86.6	30.0	45.5	68.5	84.0	38.0	87.2	62.6	58.7
+ GRPO w/ Pass-k Training	16.7	13.3	64.0	91.0	80.6	22.6	38.0	67.5	79.5	35.5	79.2	54.5	53.5
+ *ARPO*	26.7	16.7	76.8	92.8	88.0	30.8	50.5	71.5	83.0	38.0	84.8	60.1	60.0
+ *AEPO*	10.0	13.3	69.0	94.6	85.2	28.6	50.0	71.5	82.0	41.5	84.8	69.7	58.4
+ AUTOPORTFOLIO w/o scaling	23.3	23.3	75.2	93.8	86.8	28.8	52.0	72.5	85.0	45.0	90.4	68.2	62.0
+ AUTOPORTFOLIO ($N = 5$)	23.3	23.3	77.8	94.2	88.4	33.2	50.0	**76.0**	88.5	45.0	88.8	75.8	63.7
+ AUTOPORTFOLIO ($N = 10$)	26.7	23.3	81.2	95.8	91.0	35.0	50.5	75.0	90.0	**49.5**	90.4	**77.3**	65.5
Qwen2.5-7B-Instruct	20.0	16.7	77.8	93.4	89.8	35.2	25.5	60.5	70.5	30.5	71.2	41.4	52.7
+ GRPO	30.0	33.3	85.0	96.0	92.2	38.2	48.5	68.5	79.0	39.0	83.2	61.1	62.8
+ GRPO w/ Pass-k Training	26.7	30.0	86.4	94.8	92.6	40.0	41.5	60.5	76.5	36.5	80.0	60.6	60.5
+ *ARPO*	36.7	**36.7**	82.6	93.4	90.0	36.8	40.0	71.5	82.0	37.5	81.6	69.2	63.2
+ *AEPO*	33.3	30.0	86.8	94.0	93.6	39.6	45.0	73.5	85.0	39.0	80.8	76.3	64.7
+ AUTOPORTFOLIO w/o scaling	43.3	30.0	85.8	96.2	93.8	39.2	54.5	71.0	84.5	43.5	87.2	65.2	66.2
+ AUTOPORTFOLIO ($N = 5$)	**46.7**	**36.7**	87.6	**97.4**	94.0	40.6	55.5	73.5	89.5	42.5	91.2	68.2	68.6
+ AUTOPORTFOLIO ($N = 10$)	43.3	**36.7**	**91.6**	96.8	**95.2**	**42.6**	**56.0**	74.5	**90.5**	45.0	**92.8**	69.7	**69.6**

Method / Pass@1 Performance	**Mathematical Reasoning**						**Knowledge-Intensive Reasoning**						**Avg.**
	AIME24	**AIME25**	**MATH500**	**GSM8K**	**MATH**	**Omni-MATH**	**WebWalker**	**HQA**	**2Wiki.**	**MuSiQ.**	**Bamb.**	**GPQA-D**	
Llama3.1-8B-Instruct	0.0	0.0	40.0	70.4	51.4	9.8	18.5	44.5	47.5	18.0	67.2	26.3	32.8
+ GRPO	3.3	10.0	58.8	85.8	74.6	22.0	36.0	66.5	77.0	30.5	79.2	38.9	48.6
+ GRPO w/ Pass-k Training	13.3	3.3	47.0	84.0	67.4	11.8	27.5	54.5	61.5	20.5	63.2	33.3	40.6
+ *ARPO*	23.3	10.0	60.2	85.2	78.4	19.4	34.5	65.5	73.0	25.0	80.8	32.3	49.0
+ *AEPO*	6.7	10.0	52.4	84.8	69.8	19.4	35.5	63.0	73.5	30.5	76.0	42.4	47.0
+ AUTOPORTFOLIO w/o scaling	13.3	16.7	57.8	85.6	75.8	19.6	36.0	62.5	77.5	32.5	78.4	33.3	49.1
+ AUTOPORTFOLIO ($N = 3$)	13.3	13.3	63.2	88.0	76.4	21.0	36.5	63.0	76.5	35.0	80.0	41.4	50.6
+ AUTOPORTFOLIO ($N = 5$)	13.3	13.3	66.4	88.2	81.6	22.6	40.5	65.0	76.0	34.0	79.2	45.5	52.1
+ AUTOPORTFOLIO ($N = 10$)	16.7	23.3	72.6	89.8	84.0	23.2	**41.0**	64.5	**79.5**	37.5	77.6	46.0	54.6
Qwen2.5-7B-Instruct	13.3	3.3	67.2	85.6	81.8	25.2	17.5	51.0	54.0	20.0	60.0	30.3	42.4
+ GRPO	13.3	13.3	77.2	91.0	85.8	25.2	36.5	62.0	73.5	32.0	77.6	40.4	52.3
+ GRPO w/ Pass-k Training	13.3	13.3	73.4	92.4	84.4	25.0	28.5	54.0	63.5	25.5	64.8	37.9	48.0
+ *ARPO*	26.7	23.3	70.4	81.2	79.2	25.6	32.0	65.5	75.5	32.0	74.4	41.4	52.3
+ *AEPO*	16.7	23.3	76.8	91.4	88.6	26.0	33.5	59.5	71.5	29.5	74.4	**46.5**	53.1
+ AUTOPORTFOLIO w/o scaling	26.7	26.7	74.2	91.2	87.2	26.4	33.0	63.0	72.5	34.0	81.6	36.9	54.5
+ AUTOPORTFOLIO ($N = 3$)	23.3	30.0	78.6	92.8	88.8	29.8	38.5	64.0	73.0	36.5	80.0	41.4	56.4
+ AUTOPORTFOLIO ($N = 5$)	23.3	30.0	81.6	93.8	**89.2**	**31.8**	38.5	66.0	77.5	**40.0**	**83.2**	43.4	58.2
+ AUTOPORTFOLIO ($N = 10$)	**30.0**	**33.3**	**82.0**	**94.4**	89.0	31.4	40.5	**68.0**	76.5	37.5	**83.2**	46.0	**59.3**

Table 3: Comparison with pre-trained models and reasoning agents. Pass@1 and Pass@3 results on 12 challenging reasoning tasks are presented. The top two outcomes among non-muted rows are bolded and underlined.

Method / Pass@3 Performance	Mathematical Reasoning						Knowledge-Intensive Reasoning						Avg.
	AIME24	AIME25	MATH500	GSM8K	MATH	Omni-MATH	WebWalker	HQA	2Wiki.	MuSiQ.	Bamb.	GPQA-D	
(Pre-trained Models)													
Qwen2.5-72B	23.3	23.3	89.6	97.2	95.2	44.0	6.0	50.5	46.0	28.5	71.2	68.2	53.6
QwQ-32B	76.7	73.3	97.4	98.0	99.0	56.8	8.5	50.0	51.5	21.5	72.8	76.8	65.2
Qwen3-32B-thinking	83.3	73.3	96.8	97.4	99.2	56.6	9.0	42.0	50.5	23.0	66.4	78.3	64.7
DeepSeek-R1-32B	76.7	60.0	96.2	98.0	99.2	58.8	9.0	42.0	50.0	23.0	61.6	74.7	62.4
Qwen2.5-Math-7B	13.3	20.0	88.0	97.2	95.2	41.6	\	\	\	\	\	\	(59.2)
Deepseek-math-7b-rl	0.0	0.0	56.0	89.6	77.4	16.6	\	\	\	\	\	\	(39.9)
(Reasoning Agents with Tools)													
Search-r1	33.3	26.7	82.2	89.6	90.2	36.2	36.0	69.5	76.5	36.5	80.8	56.6	59.5
DeepResearcher	30.0	23.3	85.2	96.0	91.4	41.4	32.5	73.0	82.0	37.0	82.4	54.5	60.7
Tool-Light	36.7	30.0	87.0	95.8	92.6	36.2	47.0	72.0	81.5	40.5	85.6	63.1	64.0
Tool-Star	40.0	33.3	86.8	96.0	93.2	36.6	51.0	72.0	77.0	39.0	86.4	60.6	64.3
Qwen2.5-7B-Instruct	20.0	16.7	77.8	93.4	89.8	35.2	25.5	60.5	70.5	30.5	71.2	41.4	52.7
+ AUTOPORTFOLIO w/o scaling	43.3	30.0	85.8	96.2	93.8	39.2	54.5	71.0	84.5	43.5	87.2	65.2	66.2
+ AUTOPORTFOLIO ($N = 5$)	**46.7**	**36.7**	87.6	**97.4**	94.0	40.6	55.5	73.5	89.5	42.5	91.2	68.2	68.6
+ AUTOPORTFOLIO ($N = 10$)	43.3	**36.7**	**91.6**	96.8	**95.2**	**42.6**	**56.0**	**74.5**	**90.5**	**45.0**	**92.8**	**69.7**	**69.6**

Method / Pass@1 Performance	Mathematical Reasoning						Knowledge-Intensive Reasoning						Avg.
	AIME24	AIME25	MATH500	GSM8K	MATH	Omni-MATH	WebWalker	HQA	2Wiki.	MuSiQ.	Bamb.	GPQA-D	
(Pre-trained Models)													
Qwen2.5-72B	16.7	16.7	80.6	95.0	91.2	31.6	4.0	44.0	34.0	21.5	64.8	46.0	45.5
QwQ-32B	63.3	60.0	96.0	96.0	98.2	42.8	5.5	39.0	35.0	15.5	60.8	62.1	56.2
Qwen3-32B-thinking	80.0	66.7	95.4	97.0	98.6	44.2	4.5	30.5	33.0	14.5	58.4	70.2	57.8
DeepSeek-R1-32B	66.7	46.7	90.6	94.4	93.8	46.6	3.0	27.5	30.5	15.0	51.2	59.1	52.1
Qwen2.5-Math-7B	13.3	10.0	81.6	95.2	89.2	28.6	\	\	\	\	\	\	(53.0)
Deepseek-math-7b-rl	0.0	0.0	40.4	74.2	61.2	8.2	\	\	\	\	\	\	(30.7)
(Reasoning Agents with Tools)													
Search-r1	6.7	6.7	63.2	66.0	75.8	24.2	21.0	56.0	57.5	22.5	60.8	34.3	41.2
DeepResearcher	26.7	10.0	71.4	88.6	81.0	28.4	16.0	60.0	60.5	20.5	64.0	27.8	46.2
Tool-Light	26.7	20.0	77.4	91.4	87.8	23.0	32.5	59.5	65.5	28.0	75.2	41.4	52.4
Tool-Star	16.7	13.3	79.2	91.8	86.4	22.8	39.0	65.0	76.5	27.0	74.4	31.8	52.0
Qwen2.5-7B-Instruct	13.3	3.3	67.2	85.6	81.8	25.2	17.5	51.0	54.0	20.0	60.0	30.3	42.4
+ AUTOPORTFOLIO w/o scaling	26.7	26.7	74.2	91.2	87.2	26.4	33.0	63.0	72.5	34.0	81.6	36.9	54.5
+ AUTOPORTFOLIO ($N = 5$)	23.3	30.0	81.6	93.8	**89.2**	**31.8**	38.5	66.0	**77.5**	**40.0**	**83.2**	43.4	58.2
+ AUTOPORTFOLIO ($N = 10$)	**30.0**	**33.3**	**82.0**	**94.4**	89.0	31.4	**40.5**	**68.0**	76.5	37.5	**83.2**	**46.0**	**59.3**

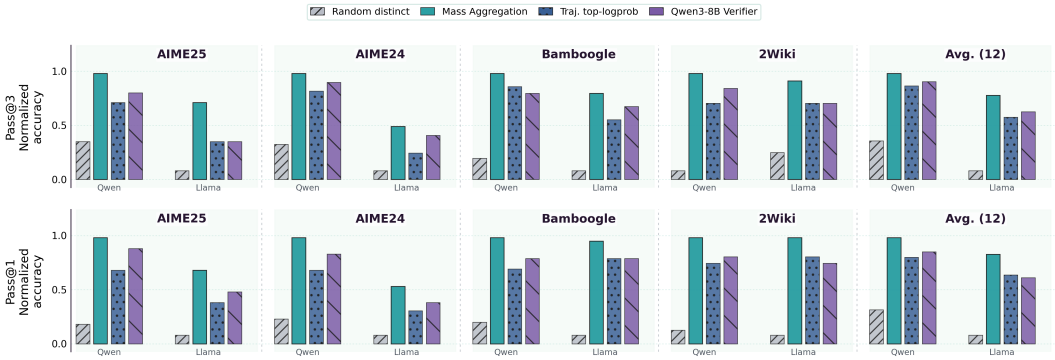

Figure 11: **Scaling-method comparison.** Within each task segment, we compare *Random*, *trajectory top-logprob*, *Qwen3-8B verifier*, and our *Mass Aggregation* for both Qwen and Llama backbones with $\lambda = 1$. Bar heights are min-max normalized within each task segment for readability. The final segment reports the 12-dataset average.

E.4.3 Comparison with Other Scaling Methods

We compare four inference-time selectors under a fixed sampling budget ($N = 32$): *Random*, trajectory top-log-probability (*traj-toplogprob*), *Qwen3-8B verifier*, and our *Mass Aggregation* (λ=1), for both Qwen and Llama backbones in Fig. 11. Unlike our lightweight Mass Aggregation, *the verifier baseline requires an additional external verification model and extra inference computation.* Mass Aggregation is generally the strongest selector across the highlighted tasks and the 12-dataset average for both Pass@3 and Pass@1. These results show that the gains of Mass Aggregation persist even against verifier-based reranking. This supports the intended mechanism: trajectory-level selectors can be distracted by individual high-confidence trajectories, while Mass Aggregation

combines confidence across trajectories reaching the same answer, making selection more reliable in high-redundancy candidate sets.

E.5 Complementary Analytical Experiments

Additional Qwen3-8B backbone diagnostic. We further compare PL-CE, ARPO, and AEPO on an additional backbone Qwen3-8B for 100 RL steps, starting from the same SFT checkpoint. Here, PL-CE generally achieves the highest training reward and held-out validation F1 (Table 4).

Table 4: A matched 100-step Qwen3-8B diagnostic. All methods start from the same SFT checkpoint. Validation F1 scores are reported separately from the test-set Pass@K results.

Method	Train reward	Validation F1@1	Validation F1@5
ARPO	0.428	0.576	0.577
AEPO	0.449	0.591	0.605
PL-CE	0.464	0.602	0.624

Evaluation uncertainty. We assess evaluation uncertainty under the end-to-end protocol of Table 1 using 64 evaluation repetitions, each drawing sub-pools of $N = 10$ trajectories per query from larger candidate pools. Matching the end-to-end protocol in Table 1, AUTOPORTFOLIO uses Mass Aggregation to retain up to $K = 3$ distinct answers, while ARPO and AEPO randomly retain $K = 3$ trajectories. For each repetition, we compute the 12-benchmark average Pass@3 difference between AUTOPORTFOLIO and each baseline. Table 5 reports the mean differences across repetitions and their confidence intervals, conditional on the trained models and candidate pools.

Table 5: End-to-end Pass@3 comparison across 64 sub-pools of $N = 10$ trajectories. Differences are AUTOPORTFOLIO minus the indicated baseline, in percentage points.

Backbone	Baseline	Mean Performance Δ Pass@3	95% CI for mean
Qwen2.5-7B	ARPO	+5.01	[4.83, 5.17]
	AEPO	+3.97	[3.76, 4.09]
Llama3.1-8B	ARPO	+5.13	[4.98, 5.29]
	AEPO	+5.40	[5.23, 5.57]

Alternative selectors. Table 6 compares selectors on the same AUTOPORTFOLIO-generated $N = 10$ pools. Meanwhile, we would like to mention that the Qwen3-8B verifier evaluation required approximately 60,000 scoring calls, over 50 million prompt tokens, and over 3 million completion tokens. Mass Aggregation uses generation log-probabilities without additional verifier-model calls.

Table 6: Additional selector comparisons on the same $N = 10$ pools for AUTOPORTFOLIO: 12-benchmark averages (%).

Backbone	Metric	Random	Qwen3-8B verifier	Mass Aggregation	Oracle@10
Qwen2.5-7B	Pass@1	54.5	57.1	59.3	73.4
	Pass@3	66.2	65.4	69.6	73.4
	Pass@5	67.0	67.6	71.7	73.4
Llama3.1-8B	Pass@1	49.1	51.9	54.6	70.0
	Pass@3	62.0	62.1	65.5	70.0
	Pass@5	64.5	65.4	68.7	70.0

F Derivation Details and Discussion

F.1 Balancing Mode Covering and Mode Seeking via Forward & Reverse CE

Setup & Notation. Fix a prompt x and a finite candidate set $\mathcal{T} = \{\tau_1, \ldots, \tau_K\}$ of full reasoning trajectories. Define the *set-conditional* distributions on $\mathcal{T}$, with indices $i \in [K] := \{1, \ldots, K\}$:

$$p_i := \frac{\pi_\theta(\tau_i|x)}{\sum_{j=1}^{K} \pi_\theta(\tau_j|x)}, \qquad q_i := \frac{\pi_{\text{ref}}(\tau_i|x)\, e^{r_i/\beta}}{\sum_{j=1}^{K} \pi_{\text{ref}}(\tau_j|x)\, e^{r_j/\beta}},$$

where $r_i := r(x, \tau_i) \in \mathbb{R}$ is a trajectory-level reward, and recall that $\beta > 0$ is the KL coefficient induced by the KL-regularized RL solution.

Suppose $\pi_\theta(\tau_i|x) > 0$ and $q_i > 0$ for all $\tau_i \in \mathcal{T}$ so that all logarithms are well-defined. Let $s_i := \log \pi_\theta(\tau_i|x) \in \mathbb{R}$ denote the (unnormalized) log-probability of τ_i under π_θ. Then $p = \mathrm{softmax}(s)$ over $\mathcal{T}$, and we have

$$\frac{\partial p_i}{\partial s_k} = p_i(\delta_{ik} - p_k), \qquad \frac{\partial \log p_i}{\partial s_k} = \delta_{ik} - p_k,$$

where δ_{ik} is the Kronecker delta, i.e., $\delta_{ik} = 1$ if $i = k$ and $\delta_{ik} = 0$ otherwise.

F.1.1 Forward CE: mode covering via $\mathrm{KL}(q\|p)$

By definition from our Subsec. 3.1.2, the forward Cross-Entropy loss is

$$\mathcal{L}_{\mathrm{fwd}}(q, p) := -\sum_{i=1}^{K} q_i \log p_i = H(q) + \mathrm{KL}(q\|p), \tag{9}$$

where $H(q)$ is constant w.r.t. θ.

Logit Gradient. Using $\frac{\partial \log p_i}{\partial s_k} = \delta_{ik} - p_k$, we can derive

$$\frac{\partial \mathcal{L}_{\mathrm{fwd}}}{\partial s_k} = -\sum_{i=1}^{K} q_i(\delta_{ik} - p_k) = p_k - q_k, \tag{10}$$

and consequently $\sum_{k=1}^{K} \partial \mathcal{L}_{\mathrm{fwd}}/\partial s_k = \sum_k (p_k - q_k) = 0$.

Why Is Forward CE Mode Covering? The key point is that $\mathcal{L}_{\mathrm{fwd}}(q, p)$ is equivalent (up to the constant $H(q)$) to $\mathrm{KL}(q\|p)$. This KL is *asymmetric*: it measures how well p explains samples drawn from q. Therefore, if q assigns non-negligible probability to multiple distinct trajectory modes (e.g., several qualitatively different but valid reasoning paths), then minimizing $\mathrm{KL}(q\|p)$ forces p to allocate probability mass to *each* q-supported mode, because p is evaluated under the weighting of q.

This yields a strong "no-mode-drop" pressure: for any index i with $q_i > 0$, if $p_i \to 0$ then

$$\mathcal{L}_{\mathrm{fwd}} = -\sum_{j=1}^{K} q_j \log p_j \geq -q_i \log p_i \xrightarrow[p_i \to 0]{} +\infty,$$

so the objective places an *infinite barrier* against eliminating any trajectory that q supports. Thus, any region with nonzero q_i but vanishing p_i receives a strong restoring force. The same intuition is visible in logit space via (10). If $p_i \ll q_i$ (in particular when q_i is non-negligible), then $p_i - q_i < 0$ will be dominated by $-q_i$, such that a gradient-descent step increases s_i and hence increases p_i. Crucially, this corrective signal does not disappear merely because p_i is already small. As long as q_i remains non-negligible, forward CE continues to push p toward covering that mode. Overall, forward CE encourages *global* matching of q, preventing collapse onto only a subset of its modes.

F.1.2 Reverse CE supports mode seeking by maximizing $\mathbb{E}_p[\log q]$

Next, recall the notation $p_i := \frac{\pi_\theta(\tau_i|x)}{\sum_{j=1}^{K} \pi_\theta(\tau_j|x)}$, $q_i := \frac{\pi_{\mathrm{ref}}(\tau_i|x)\, e^{r_i/\beta}}{\sum_{j=1}^{K} \pi_{\mathrm{ref}}(\tau_j|x)\, e^{r_j/\beta}}$, with $i \in [K]$. Let us define the reverse Cross-Entropy

$$\mathcal{L}_{\mathrm{rev}}(q, p) := -\sum_{i=1}^{K} p_i \log q_i = -\mathbb{E}_{\tau \sim p}[\log q(\tau)], \tag{11}$$

where

$$\mathcal{L}_{\mathrm{rev}}(q, p) = H(p) + \mathrm{KL}(p\|q). \tag{12}$$

In this case, minimizing reverse CE simultaneously reduces $\mathrm{KL}(p\|q)$ *and* encourages low entropy $H(p)$, which promotes concentration.

Logit Gradient. Since q is fixed w.r.t. θ, and only p depends on s, using $\frac{\partial p_i}{\partial s_k} = p_i(\delta_{ik} - p_k)$,

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{rev}}}{\partial s_k} &= -\sum_{i=1}^{K} \log q_i \cdot \frac{\partial p_i}{\partial s_k} = -\sum_{i=1}^{K} \log q_i \, p_i(\delta_{ik} - p_k) \\ &= -p_k \log q_k + p_k \sum_{i=1}^{K} p_i \log q_i \\ &= p_k\Big(\bar{\ell}_q - \log q_k\Big),\end{aligned} \tag{13}$$

where we denote $\bar{\ell}_q := \sum_{i=1}^{K} p_i \log q_i = \mathbb{E}_p[\log q(\tau)]$. Moreover, by definition, we also have $\sum_k \partial \mathcal{L}_{\text{rev}}/\partial s_k = \sum_k p_k(\bar{\ell}_q - \log q_k) = \bar{\ell}_q - \bar{\ell}_q = 0$.

Why Is Reverse CE Mode Seeking? Reverse CE can be interpreted as: *sample trajectories from p, and score them by* $\log q(\tau)$. In this case, minimizing $\mathcal{L}_{\text{rev}}$ is exactly maximizing $\mathbb{E}_p[\log q(\tau)]$, so p is encouraged to place probability mass on trajectories with the largest $\log q_i$ (equivalently, the largest q_i). Unlike forward CE, there is no barrier against assigning near-zero probability to low-q trajectories: if a trajectory has small q_i, the easiest way to avoid paying its low score $\log q_i$ is to set $p_i \approx 0$ so it is rarely sampled.

Meanwhile, this argument can be made precise via a sharp lower bound. Let $q_{\max} := \max_i q_i$. Since $\log q(\tau) \le \log q_{\max}$ pointwise, we have

$$\mathbb{E}_p[\log q(\tau)] \le \log q_{\max} \quad \Longrightarrow \quad \mathcal{L}_{\text{rev}}(q, p) \ge -\log q_{\max}. \tag{14}$$

Here, the equality holds if and only if p is supported on the argmax set $\arg\max_i q_i$ (e.g., any delta distribution δ_{i^*} with $i^* \in \arg\max_i q_i$). Therefore, *in the absence of other constraints*, reverse CE alone prefers concentrating probability mass on the best mode(s) of q, meaning that it is intrinsically *mode seeking*.

The logit-gradient geometry further explains the dynamics. Define the centered quantity

$$A_{\text{rev}}(\tau_i) := \bar{\ell}_q - \log q_i, \qquad \text{where } \bar{\ell}_q = \mathbb{E}_p[\log q(\tau)].$$

Then (13) becomes

$$\frac{\partial \mathcal{L}_{\text{rev}}}{\partial s_i} = p_i \, A_{\text{rev}}(\tau_i).$$

Thus, trajectories with $\log q_i > \bar{\ell}_q$ have negative A_{rev} and are pushed up (their logits increase), while those with $\log q_i < \bar{\ell}_q$ are pushed down. Crucially, the update is multiplied by p_i: once a trajectory receives more mass, it also receives larger absolute gradient signal, which accelerates concentration (a "rich-get-richer" effect). Conversely, trajectories with already tiny p_i contribute almost nothing to the gradient and effectively stop being updated.

Connection to the KL-regularized RL Target. On the reasoning trajectory set $\mathcal{T}$, the KL-regularized optimum has $q_i \propto \pi_{\text{ref}}(\tau_i|x)\exp(r_i/\beta)$. We can define the KL-regularized utility

$$u_i := \log \pi_{\text{ref}}(\tau_i|x) + \frac{r_i}{\beta}, \qquad q_i = \frac{e^{u_i}}{\sum_{j=1}^{K} e^{u_j}}.$$

Then, we have $\log q_i = u_i - C$ where $C = \log \sum_j e^{u_j}$, leading to

$$\bar{\ell}_q - \log q_i = \mathbb{E}_p[\log q] - \log q_i = \mathbb{E}_p[u] - u_i. \tag{15}$$

Hence the reverse-CE logit gradient becomes

$$\frac{\partial \mathcal{L}_{\text{rev}}}{\partial s_i} = p_i\big(\mathbb{E}_p[u] - u_i\big) = -p_i\big(u_i - \mathbb{E}_p[u]\big). \tag{16}$$

This alternatively shows reverse CE performs an advantage-style probability reallocation toward trajectories with above-average utility u_i (i.e., higher reward and/or higher reference likelihood).

F.1.3 PL-CE (Forward + Reverse CE): controllable coverage-concentration trade-off

As a result, we can combine the two terms as

$$\mathcal{L}_{\text{PL-CE}}(q,p) := \mathcal{L}_{\text{fwd}}(q,p) + \alpha\,\mathcal{L}_{\text{rev}}(q,p), \tag{17}$$

where $\alpha > 0$ controls the mode-covering vs. mode-seeking trade-off.

Combined Gradient (Logit Space). Combining (10) and (16), we will then have

$$\frac{\partial \mathcal{L}_{\text{PL-CE}}}{\partial s_i} = (p_i - q_i) + \alpha\, p_i\big(\mathbb{E}_p[u] - u_i\big). \tag{18}$$

Moreover, $\sum_i \partial \mathcal{L}_{\text{PL-CE}}/\partial s_i = 0$, since each component (forward or reverse CE) has zero-sum logit gradients.

Insights & Summary: How the Two Forces Interact. The forward term $(p_i - q_i)$ enforces *global support matching*: it continues to push up any trajectory with non-negligible q_i even if p_i is currently tiny, preventing premature collapse and ensuring that p does not "forget" viable reasoning modes represented in q. On the other hand, from (16), the reverse term $p_i(\mathbb{E}_p[u] - u_i)$ performs *within-support sharpening*: it reallocates probability mass among trajectories that currently have non-negligible probability under p, amplifying those with above-average utility u_i and suppressing those with below-average utility. In this case, α provides a stable knob: small α emphasizes coverage and robustness across multiple good reasoning modes, while larger α increases concentration toward the best mode(s) as measured by the KL-regularized utility u_i. As a result, PL-CE combines these two effects exactly (Eq. 18), and α controls the coverage-concentration trade-off.

F.2 Gradient Formulation Analysis and Comparisons with Pointwise Policy Gradients

This subsection analyzes the gradient structure of our *set-wise* objective and contrasts it with *pointwise* trajectory policy gradients, using GRPO as a representative baseline. Throughout, we fix a prompt x and consider a sampled candidate set (group) $\mathcal{T} = \{\tau_1, \dots, \tau_K\}$ of reasoning trajectories.

Key Findings. PL-CE is *selection-aware*: it matches the test-time goal of selecting the best trajectory from a pool by directly optimizing the *within-set* ordering on $\mathcal{T}$. Concretely, its set-wise term yields a set-conditional, zero-sum *push–pull* update $(p - q)$ in logit space, which explicitly reallocates probability mass *within* $\mathcal{T}$ and enforces set-conditional mass conservation. In contrast, GRPO assigns reward-based (z-normalized) coefficients that depend on the group only through shared reward statistics (i.e., mean and std.) and are independent of the model's current within-set probabilities p_i. As a result, GRPO is less directly optimized for the targeted within-set ranking consistency under a fixed candidate pool.

F.2.1 Setup and Notation

Define the *set-conditional* distributions on $\mathcal{T}$ (indices $i \in [K] := \{1, \dots, K\}$):

$$p_i := \frac{\pi_\theta(\tau_i|x)}{\sum_{j=1}^{K} \pi_\theta(\tau_j|x)}, \qquad q_i := \frac{\pi_{\text{ref}}(\tau_i|x)\, \exp(r_i/\beta)}{\sum_{j=1}^{K} \pi_{\text{ref}}(\tau_j|x)\, \exp(r_j/\beta)}, \tag{19}$$

where $r_i := r(x, \tau_i) \in \mathbb{R}$ is a trajectory-level reward and $\beta > 0$ is the KL coefficient from the KL-regularized RL solution. Suppose $\pi_\theta(\tau_i|x) > 0$ and $q_i > 0$ for all $\tau_i \in \mathcal{T}$ so that all logarithms are well-defined.

Let $s_i := \log \pi_\theta(\tau_i|x)$ denote the *trajectory logit* over $\mathcal{T}$. Then $p = \mathrm{softmax}(s)$ on $\mathcal{T}$ and

$$\frac{\partial p_i}{\partial s_k} = p_i(\delta_{ik} - p_k), \qquad \frac{\partial \log p_i}{\partial s_k} = \delta_{ik} - p_k, \tag{20}$$

where δ_{ik} is the Kronecker delta, i.e., $\delta_{ik} = 1$ if $i = k$ and $\delta_{ik} = 0$ otherwise.

F.2.2 Learning Objectives

Set-wise KL / CE (forward - core set-wise alignment term). For notational simplicity, we alternatively define the forward set-wise objective on the finite set as

$$\mathcal{L}_{\text{fwd}}(x, \mathcal{T}) := \mathrm{KL}\big(q \| p\big) = \sum_{i=1}^{K} q_i \log \frac{q_i}{p_i}, \tag{21}$$

which is equivalent to the forward Cross-Entropy objective $-\sum_i q_i \log p_i$, up to the entropy constant $H(q)$.

PL-CE (forward + reverse, when enabled). When including the reverse term with weight $\alpha \geq 0$, we optimize

$$\mathcal{L}_{\text{PL-CE}}(x, \mathcal{T}) := \mathcal{L}_{\text{fwd}}(x, \mathcal{T}) + \alpha\, \mathcal{L}_{\text{rev}}(x, \mathcal{T}), \qquad \mathcal{L}_{\text{rev}}(x, \mathcal{T}) := -\sum_{i=1}^{K} p_i \log q_i. \tag{22}$$

We first analyze $\mathcal{L}_{\text{fwd}}$ to expose the clean $(p - q)$ push–pull that drives ranking/selection, and then summarize the effect of the reverse term.

GRPO (with Group-relative Z-normalized Advantages). Given the same sampled group $\mathcal{T}$ with rewards $\{r_i\}_{i=1}^{K}$, define

$$\mu_r := \frac{1}{K} \sum_{j=1}^{K} r_j, \qquad \sigma_r := \sqrt{\frac{1}{K} \sum_{j=1}^{K} (r_j - \mu_r)^2}, \qquad A_i := \frac{r_i - \mu_r}{\sigma_r + \varepsilon},$$

where $\varepsilon > 0$ is a small numerical constant. An unclipped GRPO-style update (omitting PPO-style clipping for simplicity) takes the form

$$\nabla_\theta J_{\text{GRPO}}(x, \mathcal{T}) \propto \sum_{i=1}^{K} A_i\, \nabla_\theta \log \pi_\theta(\tau_i|x), \tag{23}$$

optionally plus a KL penalty term to π_{ref} to enforce a trust region.

F.2.3 Gradients of the Set-wise Objective

Forward set-wise KL: zero-sum push–pull in logit space. Using (20), the derivative of (21) with respect to logits satisfies

$$\frac{\partial \mathcal{L}_{\mathrm{fwd}}}{\partial s_i} = p_i - q_i, \qquad \sum_{i=1}^{K} \frac{\partial \mathcal{L}_{\mathrm{fwd}}}{\partial s_i} = \sum_i (p_i - q_i) = 0. \tag{24}$$

The zero-sum identity is the formal statement of *set-conditional mass conservation* on $\mathcal{T}$: probability mass is redistributed within the pool rather than globally scaled.

Recall that $s_i = \log \pi_\theta(\tau_i|x)$, then the parameter gradient becomes

$$\nabla_\theta \mathcal{L}_{\mathrm{fwd}} = \sum_{i=1}^{K} (p_i - q_i)\, \nabla_\theta \log \pi_\theta(\tau_i|x). \tag{25}$$

This update is *intrinsically set-wise*: each coefficient $(p_i - q_i)$ depends on the entire set through the softmax normalization of p.

Push–pull decomposition (interpretation). Using $\nabla_\theta \log p_i = \nabla_\theta \log \pi_\theta(\tau_i|x) - \sum_j p_j \nabla_\theta \log \pi_\theta(\tau_j|x)$, we can rewrite

$$\begin{aligned} \nabla_\theta \mathcal{L}_{\mathrm{fwd}} &= -\sum_{i=1}^{K} q_i \nabla_\theta \log p_i \\ &= \underbrace{-\sum_{i=1}^{K} q_i \nabla_\theta \log \pi_\theta(\tau_i|x)}_{\text{pull toward } q} + \underbrace{\sum_{i=1}^{K} p_i \nabla_\theta \log \pi_\theta(\tau_i|x)}_{\text{push off current mass}}. \end{aligned} \tag{26}$$

Thus, trajectories overweighted under p relative to q are pushed down, while under-allocated ones are pulled up, directly correcting relative ordering within $\mathcal{T}$.

Policy Gradient-Shaped View. Since p is a discrete distribution on $\mathcal{T}$, (25) can be equivalently written as

$$\nabla_\theta \mathcal{L}_{\mathrm{fwd}} = \mathbb{E}_{\tau \sim p}\left[\left(1 - \frac{q(\tau)}{p(\tau)}\right) \nabla_\theta \log \pi_\theta(\tau|x)\right], \qquad A^{\mathrm{PL}}(\tau) := 1 - \frac{q(\tau)}{p(\tau)}. \tag{27}$$

This highlights that PL-CE uses an *implicit* and *set-conditional* coefficient that depends on both reward (through q) and the current within-set allocation p.

F.2.4 GRPO Gradients: Pointwise Reward Weighting

GRPO assigns each sampled trajectory a coefficient determined by its (z-normalized) reward. For clarity, define the uniform empirical distribution $\hat{u}$ over the sampled group $\mathcal{T}$ by $\hat{u}(\tau_i) = 1/K$. Then (23) can be written as

$$\sum_{i=1}^{K} A_i \nabla_\theta \log \pi_\theta(\tau_i|x) = K \cdot \mathbb{E}_{\tau \sim \hat{u}}\big[A(\tau) \nabla_\theta \log \pi_\theta(\tau|x)\big].$$

Unlike (27), the expectation here is over the *empirical* distribution of sampled candidates, and *the coefficients A_i do not depend on the model's current set-conditional probabilities p_i (they depend on the group only through reward-level μ_r, σ_r)*. This difference is precisely why GRPO is not directly optimized for set-conditional mass reallocation on the pool used for ranking/selection.

F.2.5 Set-wise Coupling: GRPO vs. Forward Set-wise KL

Our forward set-wise KL (CE loss) induces *dense* coupling within $\mathcal{T}$ because the coefficients $(p_i - q_i)$ depend on the full softmax-normalized vector p. In contrast, GRPO induces only coupling through shared reward statistics (mean/std).

$$\nabla_\theta J_{\mathrm{GRPO}} \propto \sum_{i=1}^{K} A_i\, \nabla_\theta \log \pi_\theta(\tau_i|x), \qquad A_i = \frac{r_i - \mu_r}{\sigma_r + \varepsilon}. \tag{28}$$

For fixed $\mathcal{T}$ and $\{r_i\}$, the coefficients (A_i) are constant w.r.t. θ and depend on the group only through (μ_r, σ_r).

On the other hand, for Forward set-wise KL, it involves dense coupling via softmax re-normalization:

$$\nabla_\theta \mathcal{L}_{\text{fwd}} = \sum_{i=1}^{K} (p_i - q_i) \nabla_\theta \log \pi_\theta(\tau_i|x), \qquad \frac{\partial \mathcal{L}_{\text{fwd}}}{\partial s_i} = p_i - q_i. \tag{29}$$

Moreover, the Jacobian of the coefficients w.r.t. logits is

$$\frac{\partial}{\partial s_k}(p_i - q_i) = \frac{\partial p_i}{\partial s_k} = p_i(\delta_{ik} - p_k), \tag{30}$$

where δ_{ik} is the Kronecker delta, i.e., $\delta_{ik} = 1$ if $i = k$ and $\delta_{ik} = 0$ otherwise. This formulation is dense: changing one logit affects all coefficients through the normalization. This is the formal mechanism behind the set-conditional push–pull reallocation that directly targets within-set ranking/selection.

Therefore, GRPO updates are reward-driven and stabilized by group-wise normalization, but they do not explicitly implement set-conditional mass reallocation based on the model's current within-set allocation. Forward set-wise KL (and thus PL-CE) directly optimizes the relative allocation of probability mass within the candidate pool that will be used at inference time. When the reverse CE term is enabled (weight α), it contributes an additional sharpening component in logit space of the form $p_i(\bar{\ell}_q - \log q_i)$ (Subsec. F.1.2), which reallocates mass primarily among trajectories that already have non-negligible probability under p. The forward $(p - q)$ term remains responsible for global support matching and set-conditional mass conservation on $\mathcal{T}$.

G Equivalence of Cross-Entropy Loss Minimization and Plackett-Luce Ranking Optimization

In this section, we provide a detailed description and proof of Theorem 3.1 from the main body, which establishes the connection between minimizing the Cross-Entropy (CE) loss and optimizing the Plackett-Luce (PL) ranking, conditioned on a trajectory collection $\mathcal{T}$. Subsequently, we have the following Theorem G.1 that corresponds to Theorem 3.1 in the main body.

Theorem G.1 (PL ranking discrepancy bound via PL-CE)**.** *Fix a prompt $\boldsymbol{x}$ and a non-empty finite trajectory set $\mathcal{T} = \{\tau_1, \ldots, \tau_K\}$ such that $Z^* := \sum_{\tau\in\mathcal{T}} \pi^*(\tau|\boldsymbol{x}) > 0$ and $Z_\theta := \sum_{\tau\in\mathcal{T}} \pi_\theta(\tau|\boldsymbol{x}) > 0$. Suppose $\pi_{\text{ref}}(\tau|\boldsymbol{x}) > 0$ for all $\tau \in \mathcal{T}$. Define the set-conditional distributions on $\mathcal{T}$: $P(\tau) := \pi^*(\tau|\mathcal{T}, \boldsymbol{x}) = \frac{\pi^*(\tau|\boldsymbol{x})}{Z^*}$, $Q(\tau) := \pi_\theta(\tau|\mathcal{T}, \boldsymbol{x}) = \frac{\pi_\theta(\tau|\boldsymbol{x})}{Z_\theta}$. Meanwhile, denote the (set-wise) lower-mass factor $\epsilon := \min_{\tau\in\mathcal{T}} \min\{P(\tau), Q(\tau)\} > 0$. For any $\alpha \geq 0$, define the Forward-Reverse CE (PL-CE) loss over $\mathcal{T}$:*

$$\mathcal{L}^{\mathcal{T}}_{\textit{PL-CE}}(\pi^*, \pi_\theta) := \underbrace{-\sum_{\tau\in\mathcal{T}} P(\tau)\log Q(\tau)}_{\mathcal{L}^{\mathcal{T}}_{\text{fwd}}(\pi^*,\pi_\theta)} + \alpha \underbrace{\Big(-\sum_{\tau\in\mathcal{T}} Q(\tau)\log P(\tau)\Big)}_{\mathcal{L}^{\mathcal{T}}_{\text{rev}}(\pi^*,\pi_\theta)}.$$

Then, across possible permutations σ, the PL ranking discrepancy satisfies

$$\max_\sigma \Big|P_{\pi^*}(\sigma|\mathcal{T}, \boldsymbol{x}) - P_{\pi_\theta}(\sigma|\mathcal{T}, \boldsymbol{x})\Big| \leq \frac{\beta\,|\mathcal{T}|}{2\epsilon}\sqrt{2\Big(\mathcal{L}^{\mathcal{T}}_{\textit{PL-CE}}(\pi^*, \pi_\theta) - \mathcal{L}^{\mathcal{T}}_{\text{fwd}}(\pi^*, \pi^*)\Big)}. \quad (31)$$

If we have $Q(\cdot) = P(\cdot)$ on $\mathcal{T}$, the induced PL rankings $P_{\pi^*}(\cdot|\mathcal{T}, \boldsymbol{x}) = P_{\pi_\theta}(\cdot|\mathcal{T}, \boldsymbol{x})$.

Proof. The following proof is inspired by [41], while we present an improved theoretical bound, an alternative concise proof route via Lipschitzness, and an extension to our balanced PL-CE loss setting. We defer detailed discussion to Appendix G.1. First, recall that for each $\tau \in \mathcal{T}$, we define the PL score $s_\pi(\tau) := \beta \log \frac{\pi(\tau|\boldsymbol{x})}{\pi_{\text{ref}}(\tau|\boldsymbol{x})}$. Since the Plackett-Luce (PL) model is invariant to adding a constant to all scores on the same set, we can equivalently use the shifted scores

$$\tilde{s}_\pi(\tau) := s_\pi(\tau) - \beta \log Z_\pi = \beta \log \frac{\pi(\tau \mid \mathcal{T}, \boldsymbol{x})}{\pi_{\text{ref}}(\tau|\boldsymbol{x})}$$

where $Z_\pi := \sum_{\tau'\in\mathcal{T}} \pi(\tau' \mid \boldsymbol{x})$. This induces the same PL distribution on permutations of $\mathcal{T}$. Then, for any $\tau \in \mathcal{T}$, the reference cancels in score differences and we have

$$\tilde{s}_{\pi^*}(\tau) - \tilde{s}_{\pi_\theta}(\tau) = \beta \log \frac{\pi^*(\tau \mid \mathcal{T}, \boldsymbol{x})}{\pi_\theta(\tau \mid \mathcal{T}, \boldsymbol{x})} = \beta \log \frac{P(\tau)}{Q(\tau)}.$$

In this case, applying Lemma G.2 yields

$$\max_\sigma \Big|P_{\pi^*}(\sigma|\mathcal{T}, \boldsymbol{x}) - P_{\pi_\theta}(\sigma|\mathcal{T}, \boldsymbol{x})\Big| \leq \frac{\beta|\mathcal{T}|}{2\epsilon}\,\|P - Q\|_1 \leq \frac{\beta|\mathcal{T}|}{2\epsilon}\sqrt{2D_{\text{KL}}(P\|Q)}. \quad (32)$$

The first inequality uses $\epsilon := \min_{\tau\in\mathcal{T}} \min\{P(\tau), Q(\tau)\} > 0$, and the second inequality uses Pinsker's inequality $\|P - Q\|_1 \leq \sqrt{2D_{\text{KL}}(P\|Q)}$. Consequently, by the standard definition, we have

$$D_{\text{KL}}(P\|Q) = \mathcal{L}^{\mathcal{T}}_{\text{fwd}}(\pi^*, \pi_\theta) - \mathcal{L}^{\mathcal{T}}_{\text{fwd}}(\pi^*, \pi^*) = \Delta^{\mathcal{T}}_{\text{align}}(\pi^*, \pi_\theta),$$

which rewrites the KL term in (32). To obtain (31), note that $\mathcal{L}^{\mathcal{T}}_{\text{rev}}(\pi^*, \pi_\theta) = \sum_{\tau\in\mathcal{T}} Q(\tau)\big(-\log P(\tau)\big) \geq 0$, so for any $\alpha \geq 0$,

$$\Delta^{\mathcal{T}}_{\text{align}}(\pi^*, \pi_\theta) = \mathcal{L}^{\mathcal{T}}_{\text{fwd}}(\pi^*, \pi_\theta) - \mathcal{L}^{\mathcal{T}}_{\text{fwd}}(\pi^*, \pi^*) \leq \mathcal{L}^{\mathcal{T}}_{\text{PL}-\text{CE}}(\pi^*, \pi_\theta) - \mathcal{L}^{\mathcal{T}}_{\text{fwd}}(\pi^*, \pi^*).$$

Substituting completes (31) and finishes the proof. □

Lemma G.2 (PL ranking stability)**.** *Fix a prompt $\boldsymbol{x}$ and a non-empty finite trajectory set $\mathcal{T} = \{\tau_1, \ldots, \tau_K\}$ such that $Z^* = \sum_{\tau \in \mathcal{T}} \pi^*(\tau \mid \boldsymbol{x}) > 0$ and $Z_\theta = \sum_{\tau \in \mathcal{T}} \pi_\theta(\tau \mid \boldsymbol{x}) > 0$. Define the set-conditional distributions on $\mathcal{T}$, such that $P(\tau) = \pi^*(\tau \mid \mathcal{T}, \boldsymbol{x}) = \frac{\pi^*(\tau|\boldsymbol{x})}{Z^*}$, $Q(\tau) = \pi_\theta(\tau \mid \mathcal{T}, \boldsymbol{x}) = \frac{\pi_\theta(\tau|\boldsymbol{x})}{Z_\theta}$. By definition, denote the lower-mass factor $\epsilon := \min_{\tau \in \mathcal{T}} \min\{P(\tau), Q(\tau)\} > 0$. Also, denote Plackett-Luce scores $s_\pi(\tau) := \beta \log \frac{\pi(\tau|\boldsymbol{x})}{\pi_{\mathrm{ref}}(\tau|\boldsymbol{x})}$ for $\beta > 0$. Let $P_\pi(\sigma \mid \mathcal{T}, \boldsymbol{x})$ denote the PL distribution over permutations σ induced by these scores. Then,*

$$\max_\sigma \left| P_{\pi^*}(\sigma \mid \mathcal{T}, \boldsymbol{x}) - P_{\pi_\theta}(\sigma \mid \mathcal{T}, \boldsymbol{x}) \right| \le \frac{\beta |\mathcal{T}|}{2\epsilon} \|P - Q\|_1 \le \frac{\beta |\mathcal{T}|}{2\epsilon} \sqrt{2 D_{\mathrm{KL}}(P \| Q)}. \quad (33)$$

Proof. Based on the definition of PL ranking, when fixing a permutation σ of $\mathcal{T}$, we can define the remaining set as $R_i := \{\tau_{\sigma(i)}, \tau_{\sigma(i+1)}, \ldots, \tau_{\sigma(|\mathcal{T}|)}\}$ for $i = 1, \ldots, |\mathcal{T}|$. Consequently, for a policy π, we can define the stage-i categorical distribution on R_i as

$$p_i^\pi(\tau) := \frac{\exp(s_\pi(\tau))}{\sum_{\tau' \in R_i} \exp(s_\pi(\tau'))}, \qquad \tau \in R_i.$$

By PL factorization and definition, we have

$$P_\pi(\sigma \mid \mathcal{T}, \boldsymbol{x}) = \prod_{i=1}^{|\mathcal{T}|} p_i^\pi(\tau_{\sigma(i)}).$$

Since softmax is invariant to adding a constant to all logits, we can replace s_π by the shifted score

$$\tilde{s}_\pi(\tau) := s_\pi(\tau) - \beta \log \sum_{\tau' \in \mathcal{T}} \pi(\tau' \mid \boldsymbol{x}) = \beta \log \frac{\pi(\tau \mid \mathcal{T}, \boldsymbol{x})}{\pi_{\mathrm{ref}}(\tau \mid \boldsymbol{x})},$$

without changing any $p_i^\pi(\cdot)$ on R_i. Therefore, for any $\tau \in \mathcal{T}$,

$$\tilde{s}_{\pi^*}(\tau) - \tilde{s}_{\pi_\theta}(\tau) = \beta \log \frac{P(\tau)}{Q(\tau)}, \quad (34)$$

where the reference cancels in the difference. Let $a_i := p_i^{\pi^*}(\tau_{\sigma(i)})$ and $b_i := p_i^{\pi_\theta}(\tau_{\sigma(i)})$. Here, for any $\{a_i\}_{i=1}^{|\mathcal{T}|}, \{b_i\}_{i=1}^{|\mathcal{T}|}$, we have

$$\prod_{i=1}^{|\mathcal{T}|} a_i - \prod_{i=1}^{|\mathcal{T}|} b_i = \sum_{i=1}^{|\mathcal{T}|} \left(\prod_{j<i} a_j \right) (a_i - b_i) \left(\prod_{j>i} b_j \right). \quad (35)$$

Then, taking absolute values and using the triangle inequality gives

$$\left| \prod_{i=1}^{|\mathcal{T}|} a_i - \prod_{i=1}^{|\mathcal{T}|} b_i \right| \le \sum_{i=1}^{|\mathcal{T}|} \left(\prod_{j<i} |a_j| \right) |a_i - b_i| \left(\prod_{j>i} |b_j| \right).$$

In our case, since $a_i = p_i^{\pi^*}(\tau_{\sigma(i)})$ and $b_i = p_i^{\pi_\theta}(\tau_{\sigma(i)})$ lie in $[0, 1]$, the products are at most 1. We consequently have

$$\left| P_{\pi^*}(\sigma \mid \mathcal{T}, \boldsymbol{x}) - P_{\pi_\theta}(\sigma \mid \mathcal{T}, \boldsymbol{x}) \right| \le \sum_{i=1}^{|\mathcal{T}|} |a_i - b_i|. \quad (36)$$

Moreover, $|a_i - b_i| \le \|p_i^{\pi^*} - p_i^{\pi_\theta}\|_1$ since the absolute difference at one coordinate is bounded by the ℓ_1 distance. Since $p_i^\pi(\cdot)$ is obtained by applying $\mathrm{softmax}$ to the score vector $\{\tilde{s}_\pi(\tau)\}_{\tau \in R_i}$ (i.e., the logits restricted to the remaining set R_i), Lemma G.3 gives

$$\|p_i^{\pi^*} - p_i^{\pi_\theta}\|_1 \le \frac{1}{2} \sum_{\tau \in R_i} \left| \tilde{s}_{\pi^*}(\tau) - \tilde{s}_{\pi_\theta}(\tau) \right| = \frac{\beta}{2} \sum_{\tau \in R_i} \left| \log \frac{P(\tau)}{Q(\tau)} \right|. \quad (37)$$

For each $\tau \in \mathcal{T}$, by the mean value theorem for $\log(\cdot)$ on $(0, \infty)$, there exists c_τ between $P(\tau)$ and $Q(\tau)$ such that $\log P(\tau) - \log Q(\tau) = (P(\tau) - Q(\tau))/c_\tau$. Since $c_\tau \geq \min\{P(\tau), Q(\tau)\} \geq \epsilon$, we have

$$\left|\log \frac{P(\tau)}{Q(\tau)}\right| \leq \frac{|P(\tau) - Q(\tau)|}{\epsilon}.$$

Substituting into (37) gives

$$\|p_i^{\pi^*} - p_i^{\pi_\theta}\|_1 \leq \frac{\beta}{2\epsilon} \sum_{\tau \in R_i} |P(\tau) - Q(\tau)| \leq \frac{\beta}{2\epsilon} \|P - Q\|_1.$$

Therefore, $|a_i - b_i| \leq \frac{\beta}{2\epsilon}\|P - Q\|_1$ for all i. Combine with (36) to obtain

$$\left|P_{\pi^*}(\sigma \mid \mathcal{T}, \boldsymbol{x}) - P_{\pi_\theta}(\sigma \mid \mathcal{T}, \boldsymbol{x})\right| \leq \sum_{i=1}^{|\mathcal{T}|} \frac{\beta}{2\epsilon}\|P - Q\|_1 = \frac{\beta|\mathcal{T}|}{2\epsilon}\|P - Q\|_1.$$

This bound is independent of σ, so taking $\max_\sigma$ proves the first inequality in (33). The second inequality follows from Pinsker's inequality $\|P - Q\|_1 \leq \sqrt{2D_{\mathrm{KL}}(P\|Q)}$, which finishes the proof.

□

Lemma G.3 (Softmax Lipschitzness)**.** *For any $u, v \in \mathbb{R}^m$,*

$$\|\mathrm{softmax}(u) - \mathrm{softmax}(v)\|_1 \leq \frac{1}{2}\|u - v\|_1. \tag{38}$$

Proof. Let $p = \mathrm{softmax}(w)$ for some $w \in \mathbb{R}^m$. The Jacobian of softmax at w is

$$J(w)_{rk} = \frac{\partial p_r}{\partial w_k} = p_r(\delta_{rk} - p_k).$$

where δ_{rk} is the Kronecker delta, i.e., $\delta_{rk} = 1$ if $r = k$ and $\delta_{rk} = 0$ otherwise. Fix a column k. Then

$$\sum_{r=1}^{m} |J(w)_{rk}| = |p_k(1 - p_k)| + \sum_{r \neq k} |-p_r p_k| = p_k(1 - p_k) + p_k \sum_{r \neq k} p_r = 2p_k(1 - p_k) \leq \frac{1}{2},$$

since $\max_{t \in [0,1]} 2t(1 - t) = \frac{1}{2}$. Next, we can define the path $g(t) := \mathrm{softmax}(u + t(v - u))$ for $t \in [0, 1]$. By the fundamental theorem of calculus,

$$\mathrm{softmax}(v) - \mathrm{softmax}(u) = g(1) - g(0) = \int_0^1 g'(t)\, dt.$$

Moreover, by the chain rule,

$$g'(t) = J(u + t(v - u))(v - u).$$

For any matrix A and any vector x, we have the bound

$$\|Ax\|_1 = \sum_{r=1}^{m} \left|\sum_{k=1}^{m} A_{rk}x_k\right| \leq \sum_{k=1}^{m} |x_k| \sum_{r=1}^{m} |A_{rk}| \leq \Big(\max_{k \in [m]} \sum_{r=1}^{m} |A_{rk}|\Big)\|x\|_1.$$

Then, apply this inequality with $A = J(u + t(v - u))$ and $x = v - u$. Using the column-sum bound $\max_k \sum_r |J(\cdot)_{rk}| \leq \frac{1}{2}$ proved above, we obtain for every t:

$$\|g'(t)\|_1 = \|J(u + t(v - u))(v - u)\|_1 \leq \frac{1}{2}\|v - u\|_1.$$

Finally,

$$\|\mathrm{softmax}(v) - \mathrm{softmax}(u)\|_1 \leq \int_0^1 \|g'(t)\|_1\, dt \leq \int_0^1 \frac{1}{2}\|v - u\|_1\, dt = \frac{1}{2}\|u - v\|_1,$$

which finishes the proof of this lemma. □

G.1 Further Insights

Here, our Theorem G.1 makes explicit that set-conditional alignment on the sampled pool $\mathcal{T}$ is sufficient to control the induced PL ranking behavior on that same pool. Meanwhile, note that it does not guarantee monotonic improvement of the complete sample-then-select objective, which can also depend on candidate-generation coverage and generalization across sampled pools. In particular, the PL ranking discrepancy is upper bounded by a constant factor times $\sqrt{D_{\mathrm{KL}}(P\|Q)}$, where $P(\cdot) = \pi^*(\cdot|\mathcal{T}, \boldsymbol{x})$ and $Q(\cdot) = \pi_\theta(\cdot|\mathcal{T}, \boldsymbol{x})$. Therefore, minimizing the forward Cross-Entropy on $\mathcal{T}$ (equivalently minimizing $D_{\mathrm{KL}}(P\|Q)$) is a principled objective for recovering the optimal PL ranking used at inference time. This forward-CE and PL-ranking connection is also developed in [41]. Our result strengthens the alignment with the *actual* training objective used in this work: the bi-directional PL-CE loss $\mathcal{L}^{\mathcal{T}}_{\text{PL-CE}} = \mathcal{L}^{\mathcal{T}}_{\text{fwd}} + \alpha\, \mathcal{L}^{\mathcal{T}}_{\text{rev}}$. Crucially, for any $\alpha \geq 0$ and any iterate, we have $\mathcal{L}^{\mathcal{T}}_{\text{fwd}} \leq \mathcal{L}^{\mathcal{T}}_{\text{PL-CE}}$. Thus, a small PL-CE value directly certifies a small forward CE (hence a small $D_{\mathrm{KL}}(P\|Q)$), and by Theorem G.1 it guarantees a small PL ranking discrepancy on the same candidate pool $\mathcal{T}$. Moreover, the reverse CE term provides an additional *two-sided* control: since $D_{\mathrm{KL}}(Q\|P) = \mathcal{L}^{\mathcal{T}}_{\text{rev}} - H(Q) \leq \mathcal{L}^{\mathcal{T}}_{\text{rev}}$, we can tighten the intermediate variation step by using $\|P - Q\|_1 \leq \min\{\sqrt{2D_{\mathrm{KL}}(P\|Q)},\ \sqrt{2D_{\mathrm{KL}}(Q\|P)}\}$, which yields a strictly smaller RHS whenever the reverse term is also small (i.e., it improves the effective constant inside the square-root by taking the better of the two directions).

This theoretical bound also highlights a standard regularity requirement through the set-wise lower-mass factor $\epsilon := \min_{\tau\in\mathcal{T}} \min\{P(\tau), Q(\tau)\} > 0$. This condition rules out degenerate cases where either distribution assigns zero or nearly zero probability to some $\tau \in \mathcal{T}$, and is analogous to concentrability conditions commonly used in RL analysis (e.g., [7, 25]). In this case, a sufficient condition is a bounded density ratio on the set $\mathcal{T}$, e.g., $\left\|\frac{P(\tau)}{Q(\tau)}\right\|_\infty \leq \epsilon'$ with $\epsilon' \geq 1$, which implies $Q(\tau) \geq P(\tau)/\epsilon'$. Hence, $\epsilon \geq \min_\tau \min\{P(\tau), P(\tau)/\epsilon'\} = \frac{1}{\epsilon'} \min_\tau P(\tau)$. This removes explicit dependence on π_θ from the prefactor (at the cost of introducing a problem-dependent constant), which is also a common refinement in ranking analysis [41]. In practice, since $\mathcal{T}$ is constructed by our trajectory generation procedure, we can encourage non-vanishing set-conditional mass on the π_θ side (i.e., prevent extremely small $Q(\tau)$ on the sampled set), by injecting mild exploration or mixture sampling (e.g., ϵ-greedy [14]). This stabilizes set-wise optimization on $\mathcal{T}$ and supports the conditions under which Eq. (31) is meaningful.

H Case Study on Agentic Reasoning Process

To complement our quantitative evaluation, we present a qualitative case study on OMNI-MATH that examines the agentic reasoning trajectories produced by AUTOPORTFOLIO in comparison with AEPO. We report the three trajectories with the highest confidence under the corresponding method.

In the examples shown, AUTOPORTFOLIO's top three most confident rollouts all produce the correct final answer, whereas AEPO's top three most confident rollouts are consistently incorrect. The baseline failures are not primarily attributable to minor algebraic or computational slips; instead, they frequently originate from an early misinterpretation of the problem statement. We explicitly annotate the earliest conceptual mistake in the AEPO trajectories to highlight that the error arises at the level of problem understanding (Cases 4 and 5), alongside downstream execution (Case 6).

Although these demonstrations are not intended as a statistical substitute for the main evaluation, they provide concrete qualitative evidence that AUTOPORTFOLIO's intrinsic ranking mechanism is meaningfully correlated with reasoning quality. In particular, the method tends to assign high confidence to trajectories that reach the correct final answer, while the baseline can produce high-confidence trajectories that are systematically derailed by an early conceptual error. Taken together, these case studies support the conclusion that explicitly training for set-conditional alignment can improve the reliability of agentic reasoning in settings where multiple rollouts need to be generated and prioritized.

Case Study on Omni-MATH

Input Prompt:
You are a helpful assistant that can solve the given question using the format: `<think> ... </think> <python> ... </python> <result> ... </result> <answer> ... </answer>`
Three noncollinear points and a line ℓ are given in the plane. Suppose no two of the points lie on a line parallel to ℓ (or ℓ itself). There are exactly n lines perpendicular to ℓ with the following property: the three circles with centers at the given points and tangent to the line all concur at some point. Find all possible values of n.

AUTOPORTFOLIO Response:

<think> For a line $m \perp \ell$, the circle centered at P tangent to m has radius $r_P = d(P, m)$. WLOG rotate so ℓ is horizontal; then m is vertical ($x = t$). Using $r_P^2 = (p_x - t)^2$, the three circle equations share a common point iff the induced "difference-of-circles" linear constraints yield a feasible intersection; under the non-degeneracy assumptions, this happens for exactly one m, hence $n = 1$.
</think>

Sanity-check on a concrete non-degenerate instance via symbolic elimination: solve two linear (difference) equations for (X, Y) as functions of t, then enforce one circle equation to count valid t. **<tool>**Code Interpreter (CI)**</tool>** **<python>**

```
import sympy as sp

# Example (noncollinear; distinct x-projections w.r.t. ell):
# ell: y=0, m: x=t
A = sp.Point(0, 1); B = sp.Point(2, 3); C = sp.Point(5, -2)
t, X, Y = sp.symbols('t X Y')

def eq(P):
  px, py = P.x, P.y
  return (X-px)**2 + (Y-py)**2 - (px-t)**2   # r_P^2 = (px-t)^2

# Eliminate (X,Y) using difference-of-circles (linear in X,Y)
lin1 = sp.expand(eq(B) - eq(A))
lin2 = sp.expand(eq(C) - eq(A))
solXY = sp.solve([lin1, lin2], [X, Y], dict=True)[0]

# Residual condition in t (must satisfy one circle equation)
res = sp.expand(eq(A).subs(solXY))
poly = sp.Poly(sp.factor(res), t)

roots = poly.nroots()
real_roots = [r for r in roots if abs(complex(r).imag) < 1e-7]
print("n=",len(real_roots),";t$\approx$",
        [round(complex(r).real, 4) for r in real_roots])
```

</python>
<result>
(Example run)
n = 1 ; t ≈ [...]
</result>
<answer> Therefore, the only possible value is `\boxed{1}`. **</answer>**
Trajectory logprob: `-97.05`

Case Study on Omni-MATH

Input Prompt:
You are a helpful assistant that can solve the given question ... For example, <think> This is the reasoning process. </think> <python> python code here </python> <result> python interpreter result here </result> <think> This is the reasoning process. </think> <answer> The final answer is \boxed{answer here} </answer> ... Three noncollinear points and a line ℓ are given in the plane. Suppose no two of the points lie on a line parallel to ℓ (or ℓ itself). There are exactly n lines perpendicular to ℓ such that the three circles centered at the points and tangent to the line all concur at some point. Find all possible values of n.

AUTOPORTFOLIO Response:

<think> Key reduction. WLOG set ℓ horizontal, so any $m \perp \ell$ is vertical: $m: x = t$. For $P = (p_x, p_y)$, tangency radius is $r_P = d(P, m) = |p_x - t|$. Concurrency means there exists (X, Y) satisfying three circle equations; subtracting pairs cancels quadratics and yields *two linear equations in* (X, Y) (affine in t), hence at most one candidate $(X(t), Y(t))$ per t. Checking it against a remaining circle gives one scalar constraint in t; under non-degeneracy, it has exactly one solution. **Thus** $n = 1$ and the answer is `\boxed{1}`. **</think>**

A compact sanity-check: pick a concrete configuration, compute the candidate point by solving the two linear "difference-of-circles" equations for each t, then count roots of the residual. **<tool>**Code Interpreter (CI)**</tool>**
<python>

```
import numpy as np
A=np.array([0.,1.]); B=np.array([2.,3.]); C=np.array([5.,-2.])
# nondegenerate

def XY(t):
  def row(P,Q):
    # after canceling r_Q^2-r_P^2
    dx,dy=(Q-P); rhs=(Q[1]**2-P[1]**2)+2*t*dx
    return np.array([2*dx,2*dy]), rhs
  r1,b1=row(A,B); r2,b2=row(A,C)
  M=np.vstack([r1,r2]); v=np.array([b1,b2])
  if abs(np.linalg.det(M))<1e-10: return None
  return np.linalg.solve(M,v)

def f(t):
  z=XY(t)
  if z is None: return np.nan
  X,Y=z; rA2=(A[0]-t)**2
  return (X-A[0])**2+(Y-A[1])**2-rA2

# scan for sign-changes + bisection refine (details omitted)
ts=np.linspace(-5,10,2001); vals=np.array([f(t) for t in ts])
roots=[]
for i in range(len(ts)-1):
  if np.isfinite(vals[i]) and np.isfinite(vals[i+1]) \
        and vals[i]*vals[i+1]<0:
    lo,hi=ts[i],ts[i+1]
    for _ in range(35):  # ...
      mid=(lo+hi)/2
      if f(lo)*f(mid)<=0: hi=mid
      else: lo=mid
    roots.append((lo+hi)/2)

# dedup + report
uniq=[]
for r in sorted(roots):
  if not uniq or abs(r-uniq[-1])>1e-3: uniq.append(r)
print("n=", len(uniq), "t$\approx$", [round(r,4) for r in uniq])
```

</python>
<result>
(Example run)
n = 1 t≈ [. . .]
</result>
Final: `\boxed{1}`.
Trajectory logprob: `-64.22`

Case Study on Omni-MATH

Input Prompt:
You are a helpful assistant that can solve the given question using the format: `<think> ... </think> <python> ... </python> <result> ... </result> <think> ... </think> <answer> ... </answer>`
Three noncollinear points and a line ℓ are given in the plane. Suppose no two of the points lie on a line parallel to ℓ (or ℓ itself). There are exactly n lines perpendicular to ℓ with the following property: the three circles with centers at the given points and tangent to the line all concur at some point. Find all possible values of n.

AUTOPORTFOLIO Response:
<think> We have three noncollinear points A, B, C and a line ℓ. For a circle centered at P tangent to a line, its radius equals the perpendicular distance from P to that line. We seek the number of lines perpendicular to ℓ such that the three induced circles (centered at A, B, C with those radii) intersect at a common point.
A direct symbolic solve can be brittle, but the geometry suggests uniqueness under the given non-degeneracy assumptions.
</think>
<python>

```
import sympy as sp

# Symbolic illustration attempt (kept compact; details omitted):
x, y = sp.symbols('x␣y')
# x-coordinates of three centers (schematic)
a, b, c = sp.symbols('a␣b␣c')
m, p = sp.symbols('m␣p')            # line ell: y = m x + p

# Distance from (u,0) to ell is |m u + p| / sqrt(m^2+1)
def dist(u):
  return sp.Abs(m*u + p) / sp.sqrt(m**2 + 1)

# Circles centered at (a,0), (b,0), (c,0) with radii dist(*)
CA = (x-a)**2 + y**2 - dist(a)**2
CB = (x-b)**2 + y**2 - dist(b)**2
CC = (x-c)**2 + y**2 - dist(c)**2

# Solver may return empty here due to abs/branching complexity.
sol = sp.solve([CA, CB, CC], (x, y), dict=True)
print(len(sol))
```

</python>
<result>
0
</result>
<think> The symbolic system can fail (e.g., due to absolute values and branching), but the geometric structure is clearer: for each candidate line $m \perp \ell$, the three radii are fixed as distances-to-m. Under noncollinearity and the condition that no two points share the same projection along ℓ (i.e., no line parallel to ℓ through two points), the induced concurrency configuration occurs for exactly one such perpendicular line. Hence $n = 1$. **</think>**
<answer> Thus, the only possible value of n is `\boxed{1}`. **</answer>**
Trajectory logprob: `-97.16`

Case Study on Omni-MATH

Input Prompt:
You are a helpful assistant that can solve the given question step by step with the help of the python interpreter tool ... For example, <think> This is the reasoning process. </think> <python> python code here </python> <result> python interpreter result here </result> <think> This is the reasoning process. </think> <answer> The final answer is \boxed{answer here} </answer> ...

Three noncollinear points and a line ℓ are given in the plane. Suppose no two of the points lie on a line parallel to ℓ (or ℓ itself). There are exactly n lines perpendicular to ℓ with the following property: the three circles with centers at the given points and tangent to the line all concur at some point. Find all possible values of n.

AEPO Response:

<think> To solve this problem, we need to understand the geometric configuration and use natural language reasoning combined with some algebraic manipulation.
Given three noncollinear points A, B, and C and a line ℓ, we are asked to find the number of lines perpendicular to ℓ such that the three circles with centers at A, B, and C and tangent to ℓ all concur at some point. **</think>**

Mistake starts here: misread "tangent to the line" as tangent to ℓ (should be tangent to the varying $m \perp \ell$)

Wrong step (from the model):
"Each circle centered at A, B, C and tangent to the line ℓ will have its radius equal to the perpendicular distance from the center to the line ℓ."

We can illustrate and sanity-check the claimed geometric property with a simple symbolic/numeric construction. A code execution environment is suitable for quickly verifying the number of candidate lines under a convenient coordinate setup.
<tool>Code Interpreter (CI)**</tool>**

<python>

```
...
# Define three points A, B, C (non-collinear)
A = (0, 0)
B = (1, 1)
C = (2, 0)

# Function to calculate the perpendicular distance from a point to a line
def perpendicular_distance(point, line):
    x0, y0 = point
    a, b = line.coeff(x), line.coeff(y)
    return abs(a*x0 + b*y0) / sp.sqrt(a**2 + b**2)
...
# Check if there are any other lines perpendicular to l
# that could satisfy the condition
# We know there should be exactly 2 such lines (one on each side of l)
possible_lines = [y - dist_A, y + dist_A]

# Print the number of such lines
print(len(possible_lines))
```

</python>
<result>
2 **</result>**
<answer> Final: The number of lines perpendicular to ℓ such that the three circles with centers at the given points and tangent to the line all concur at some point is `\boxed{2}`. **</answer>**

Trajectory logprob: `-143.20`

Case Study on Omni-MATH

Input Prompt:
You are a helpful assistant that can solve the given question step by step with the help of the python interpreter tool ... For example, <think> This is the reasoning process. </think> <python> python code here </python> <result> python interpreter result here </result> <think> This is the reasoning process. </think> <answer> The final answer is \boxed{answer here} </answer> ...

Three noncollinear points and a line ℓ are given in the plane. Suppose no two of the points lie on a line parallel to ℓ (or ℓ itself). There are exactly n lines perpendicular to ℓ with the following property: the three circles with centers at the given points and tangent to the line all concur at some point. Find all possible values of n.

AEPO Response: **<think>** To solve this problem, we need to understand the geometric configuration and use natural language reasoning combined with algebraic verification.
Given three noncollinear points A, B, and C and a line ℓ, we need to find the number of lines perpendicular to ℓ such that the three circles centered at A, B, and C and tangent to ℓ all concur at some point. Step-by-Step Reasoning:
1. **Understanding the Problem:**

- We have three points A, B, and C which are not collinear.
- We have a line ℓ.
- We need to find lines perpendicular to ℓ such that the circles centered at A, B, and C and tangent to ℓ intersect at a common point.

Mistake starts here: misread "tangent to the line" as tangent to ℓ (should be tangent to the varying $m \perp \ell$)

Wrong step (from the model):
"Each circle centered at A, B, and C and tangent to ℓ will touch ℓ at exactly one point each."

...
Thus, the number of such lines is 2. Let's confirm this with a simple Python code using sympy to verify the geometric properties. **</think>**
We can sanity-check the claimed conclusion with a simple symbolic construction under a convenient coordinate setup.
<tool>Code Interpreter (CI)**</tool>**

<python>

```
...

# Define three points A, B, C
A = (sp.Symbol('Ax'), sp.Symbol('Ay'))
B = (sp.Symbol('Bx'), sp.Symbol('By'))
C = (sp.Symbol('Cx'), sp.Symbol('Cy'))

...

# The points of tangency of the circles centered at A, B, C
# with the line y = c
tangent_A = (A[0], c)
tangent_B = (B[0], c)
tangent_C = (C[0], c)

# There are exactly two such lines perpendicular to the given line l
# that satisfy the condition
n = 2

print(n)
```

</python>
<result>
2 **</result>**
<answer>Final: `\boxed{2}`**.</answer>**

Trajectory logprob: `-147.93`

Case Study on Omni-MATH

Input Prompt:
You are given a problem and a set of tools. Solve the problem step by step by selecting the proper tools to use during your thinking.

Here is the name and usage instructions for the toolset, where you will select the appropriate tools to use.
Tool 1: Code Interpreter (CI) ...
Tool 2: OCR ...
Tool 3: SearchAPI ...
...
Three noncollinear points and a line ℓ are given in the plane. Suppose no two of the points lie on a line parallel to ℓ (or ℓ itself). There are exactly n lines perpendicular to ℓ with the following property: the three circles with centers at the given points and tangent to the line all concur at some point. Find all possible values of n.

AEPO Response:

<think> To solve this problem, we need to understand the geometric configuration and the conditions given. We are given three noncollinear points A, B, and C in a plane, along with a line ℓ. We need to find the number of lines perpendicular to ℓ such that the three circles centered at A, B, and C and tangent to the line ℓ all concur at some point.
Let's break down the problem:

1. **Understanding the Problem:**
   - Each circle is centered at one of the points A, B, or C and is tangent to the line ℓ.
   - The distance from each center to the line ℓ must be equal to the radius of the respective circle.
   - We need to find how many lines perpendicular to ℓ satisfy the condition that these three circles concur at some point.

Mistake starts here: misread "tangent to the line" as tangent to ℓ (should be tangent to the varying $m \perp \ell$)

Wrong step (from the model):
"Each circle is centered at one of the points A, B, or C and is tangent to the line ℓ."

...

So, the number of such lines is $n = 6$.
Let's confirm this with a simple Python code using sympy to simulate the geometric conditions, although the problem is more about geometric reasoning than symbolic computation. **</think>**
We can quickly validate the claimed count by writing a small symbolic/numeric checker; a code execution environment is suitable for this purpose. **<tool>**Code Interpreter (CI)**</tool>**
<python>

```
...
# Function to check if three circles centered at A, B, C
# and tangent to line l are concurrent
def are_circles_concurrent(Ax, Ay, Bx, By, Cx, Cy, dx, dy):
    # Calculate the distances from A, B, C to the line l
    d_A = sp.sqrt((Ax - dx)**2 + (Ay - dy)**2)
    d_B = sp.sqrt((Bx - dx)**2 + (By - dy)**2)
    d_C = sp.sqrt((Cx - dx)**2 + (Cy - dy)**2)

    ...
    return 6

...

# Count the number of valid perpendicular lines
n = are_circles_concurrent(*example_points[0],
*example_points[1], *example_points[2], *line_direction)
print(n)
```

</python>
<result>
6 **</result>**
<answer>Final: \boxed{6}.**</answer>**

Trajectory logprob: -173.46

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Our claims are properly summarized in our Introduction section.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Please see Appendix C.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: We provide detailed discussion in Appendix G.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provided detailed experiment settings in the main body and Appendix. We will release the source code upon paper acceptance.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We will release the source code upon paper acceptance. We have also included detailed descriptions, such as hyper-parameter configurations, for our experiments in the Appendix.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We have provided detailed experiment settings in the Appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Due to the high cost of agentic RL training and third-party search-API queries, we did not run multiple independent training seeds. Instead, we report 95% confidence intervals for mean Pass@3 differences across 64 fixed-budget evaluation sub-pools. These intervals are conditional on the trained models and candidate banks. We did not run independent training-seed repetitions. We also support the credibility of the main claims through comprehensive evaluation across 12 diverse benchmarks, two backbones, multiple inference budgets up to Pass@32, Oracle@N upper-bound analysis, and selection/aggregation ablations that separate candidate coverage from within-pool ranking.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We have provided detailed experiment resources in the Appendix.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We conform with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We include Broader Impacts in Appendix B.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: N/A

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We have properly cited the asset sources.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: N/A

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: N/A

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: N/A

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: Please see Appendix B.

    Guidelines:

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