

J4R: Learning to Judge with Equivalent Initial State Group Relative Policy Optimization

Austin Xu*, Yilun Zhou*, Xuan-Phi Nguyen, Caiming Xiong*, Shafiq Joty

Salesforce AI Research

sjoty@salesforce.com

 [GitHub](#)

 [ReasoningJudgeBench](#)

Abstract

To keep pace with the increasing velocity of large language models (LLM) development, model output evaluation has transitioned away from time-consuming human evaluation to automatic evaluation, where LLMs themselves are tasked with assessing and critiquing other model outputs. LLM-as-judge models are a class of generative evaluators that excel in evaluating relatively simple domains, like chat quality, but struggle in reasoning intensive domains where model responses contain more substantive and challenging content. To remedy existing judge shortcomings, we explore training judges with reinforcement learning (RL). We make three key contributions: (1) We propose the Equivalent Initial State Group Relative Policy Optimization (EIS-GRPO) algorithm, which allows us to train our judge to be robust to positional biases that arise in more complex evaluation settings. (2) We introduce ReasoningJudgeBench, a benchmark that evaluates judges in diverse reasoning settings not covered by prior work. (3) We train Judge for Reasoning (J4R), a 7B judge trained with EIS-GRPO that outperforms GPT-4o and the next best small judge by 6.7% and 9%, matching or exceeding the performance of larger GRPO-trained judges on both JudgeBench and ReasoningJudgeBench.

1 Introduction

Large language models (LLMs) and LLM-powered agentic systems have been tasked with solving increasingly difficult problems that require strong reasoning abilities (Ke et al., 2025). As LLMs are deployed for reasoning intensive-tasks, evaluation methods have come under mounting pressure to efficiently and accurately evaluate and critique LLM outputs, especially when the outputs are not easily verifiable with rules. While human evaluation remains the gold-standard in such cases, automatic

evaluation approaches, whether it be reward models (RMs) or LLM-as-judges, have been deployed as a more scalable alternative. However, these approaches have shortcomings (Cemri et al., 2025): Current verifiers tend to lock onto stylistic details over substance (Feuer et al., 2024; Liu et al., 2024; Zeng et al., 2023). RMs are incapable of explaining their decisions, while judge models are prone to various biases such as positional (Li et al., 2023; Wang et al., 2023a) and length bias (Zeng et al., 2023; Park et al., 2024). In all, these drawbacks lead to evaluators that struggle to evaluate in reasoning settings (Tan et al., 2024; Zhou et al., 2025).

This work explores using reinforcement learning from verifiable rewards (RLVR) (Lambert et al., 2024; Shao et al., 2024; Guo et al., 2025) to improve the evaluation ability of LLM-as-judge models in reasoning problems. Automatic evaluation is a natural setting for RLVR: Reinforcement learning (RL) allows the model to learn how to generate chain-of-thought (CoT) critiques without supervision from distilled CoT critiques, which are commonly used in judge model training (e.g., (Kim et al., 2023, 2024b; Wang et al., 2024a; Ye et al., 2024)). Furthermore, automatic evaluation outputs have a limited set of final outcomes (e.g., “A” or “B” for pairwise comparisons, 1-5 for single ratings), making them naturally “verifiable”.

Before simply applying RL algorithms (Shao et al., 2024; Guo et al., 2025) with judge data, we take a step back and carefully analyze how judges fail in reasoning settings. Our analysis in Sec. 4 shows that as task difficulty grows, existing judges, including those trained with RL, suffer from increasing *positional inconsistency*, where swapping the order of the candidates does not lead to an equivalent swap in judgment. This inconsistency indicates that the judge is randomly guessing, rather than substantively assessing. To remedy this, we propose Equivalent Initial State Group Relative Policy Optimization (EIS-GRPO), which explic-

*Work done at Salesforce AI Research

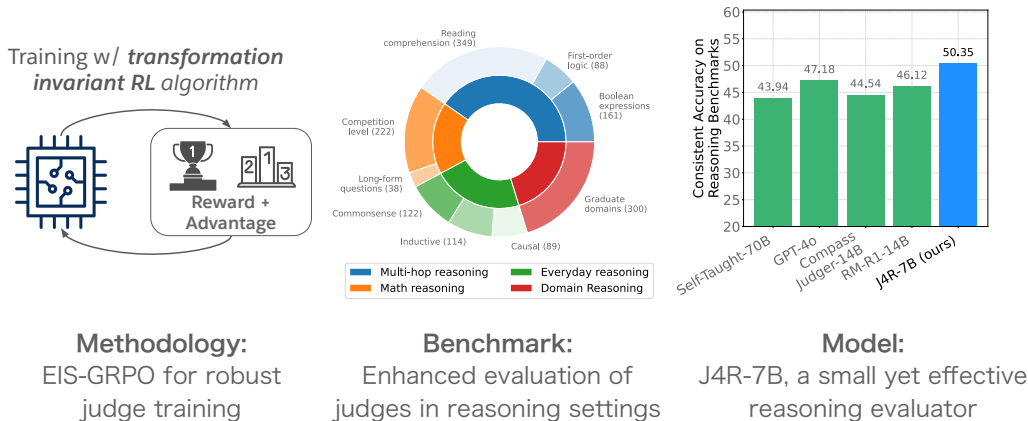


Figure 1: Our three contributions: (1) EIS-GRPO, a RL algorithm for training positionally robust judge models. (2) ReasoningJudgeBench, a benchmark of 1,483 pairwise samples across diverse and underexplored reasoning settings. (3) J4R-7B, a lightweight yet high-performing judge specifically for reasoning evaluation.

itly trains the model to treat equivalent transformations of the same initial state (i.e., input context) as equivalent, instilling consistency and robustness in judges with zero additional training overhead. Using EIS-GRPO, we train Judge for Reasoning (J4R), a 7B LLM judge that specializes in evaluating reasoning. J4R outperforms GPT-4o by 6.7% and approaches the performance of 32B RL-trained judges on a reasoning-focused judge benchmarks.

In our evaluation, we find existing benchmarks of reasoning judges lack breadth. Notably, JudgeBench (Tan et al., 2024) consists only of 350 pairwise samples sourced from MMLU-Pro (Wang et al., 2024c), LiveBench (White et al., 2024), and LiveCodeBench (Jain et al., 2024), missing coverage of different, uniquely challenging reasoning tasks, like inductive or commonsense reasoning. This leads us to create ReasoningJudgeBench which covers new, challenging types of reasoning. As shown in Fig. 1, our contributions are threefold:

- **Methodology:** We propose Equivalent Initial State GRPO, a simple but effective data augmentation method for GRPO that produces positionally robust judge models.
- **Benchmark:** We create ReasoningJudgeBench, a benchmark of 1,483 challenging pairwise samples that is 4x larger and more diverse than existing reasoning-focused judge benchmarks.
- **Model and Analysis:** We train J4R-CJ-7B which beats the next best ≤ 14 B judge by 13% and 9% on JudgeBench and ReasoningJudgeBench and matches larger judge models. We quantify the efficacy of EIS-GRPO and the benefits of inference-time scaling, among other analyses.

2 Background and related work

Automatic evaluation. Advances in reasoning have largely focused on settings where answers are easily verifiable (e.g., math). However, many complex problems, such as long-form report generation, cannot be graded solely on answer correctness. As a result, strong automatic evaluators are needed for such reasoning tasks. Automatic evaluation methods broadly fall under two paradigms: reward models (RMs) (Zhong et al., 2025), which are non-generative evaluators that output scalar scores, and LLM-as-judge (Gu et al., 2024), which are generative. The latter has gained popularity, with a line of work training specialized judge models using supervised finetuning (SFT) (Li et al., 2023; Kim et al., 2023, 2024a), preference optimization (Wang et al., 2024a; Ye et al., 2024; Hu et al., 2024; Wang et al., 2024b; Saha et al., 2025), and hybrid objectives (Zhang et al., 2024; Mahan et al., 2024).

Only recently have methods for improving reasoning been applied to judges, with (Liu et al., 2025c) exploring inference-time scaling. More recent works train judge models with RL (Chen et al., 2025a,b; Whitehouse et al., 2025; Chan et al., 2025; Hong et al., 2025), with a focus on analyzing the effects of training pipeline, data, and reward design; Our work can be seen as *complementary* to newer RL-training approaches: We propose a novel training *methodology*, EIS-GRPO, which was developed after analyzing key failure modes of judges in reasoning settings. Furthermore, our analysis in Sec. 4 reveals that *using RL training alone is insufficient for evaluating in more difficult reasoning domains*. Concretely, when evaluating

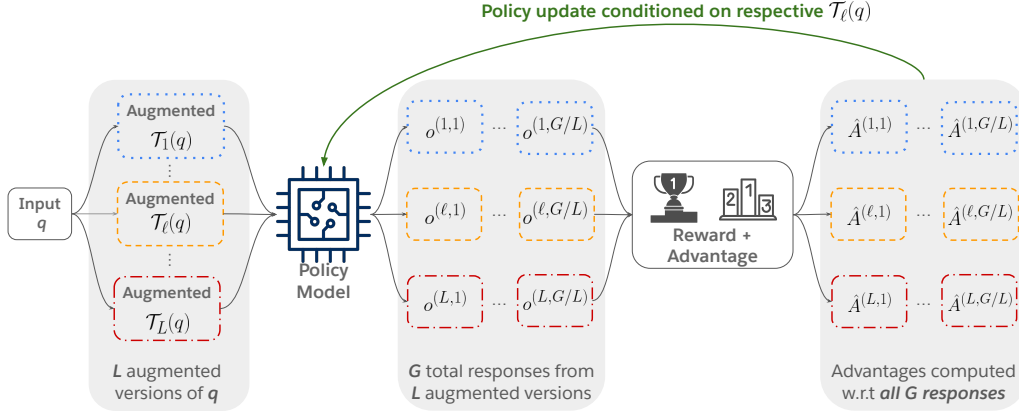


Figure 2: Overview of EIS-GRPO: Each input q is first transformed via \mathcal{T}_ℓ to a *substantively equivalent* version of q and used to sample a subgroup of responses. Responses are rewarded with outcome verification and advantages are computed jointly across subgroups and within subgroups before being used to update the policy model with respect to the transformed input. Joint subgroup advantage computation teaches the model to treat all transformed inputs as *equivalent*, training the model to be robust to non-substantive transformations of the input.

challenging samples, RL-trained judges still exhibit random guessing behavior, which manifests as pairwise *inconsistency*: When the order of responses is switched, the judge’s chosen response switches. As such, in contrast to past advances in judge training, which simply apply the latest training algorithms (DPO, GRPO, etc.) for judges, we propose a methodological advancement: To our knowledge, our work is the *first RL-based training algorithm designed with training judges in mind*.

3 Equivalent Initial State GRPO

Here we present Equivalent Initial State GRPO, a policy gradient algorithm for training models to be robust to input transformations. Our presentation here is in a general manner, and we describe specific application to judge model training in Sec. 4. While what we present is specifically motivated via GRPO, the core ideas are generally applicable other algorithms, like RLOO (Kool et al., 2019).

3.1 Background: GRPO

We use subscripts t to denote token index, superscripts (i) to denote sample index (i.e., if multiple outputs are generated for the same input).

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is a policy gradient algorithm that constrains policy updates, preventing large updates and resulting in more stabilized training. When applied by earlier works (Ouyang et al., 2022; Stiennon et al., 2020) for finetuning LLMs, given a question (prompt) q , each output token o_t is modeled as a separate *action*, with $(q, o_{<t})$ as the *state*. Specifi-

cally, given a dataset $\mathcal{D} = \{q\}$ of questions q , the LLM is optimized with the PPO objective

$$\mathcal{L}_{\text{PPO}}(\theta) = \mathbb{E}_{\substack{q \sim \mathcal{D}, \\ o \sim \pi_{\theta_{\text{old}}}(\cdot|q)}} \left[\frac{1}{|o|} \sum_{t=1}^{|o|} \left(\min \left\{ r_t(\theta) \hat{A}_t, \right. \right. \right. \\ \left. \left. \left. \text{clip}(r_t(\theta); \epsilon) \hat{A}_t \right\} - \beta D_{\text{KL}}(\pi_\theta || \pi_{\text{ref}}) \right) \right], \quad (1)$$

where ϵ is the clipping value that controls the policy update range, \hat{A}_t is the *advantage* estimate for token t , D_{KL} is the KL divergence, which discourages large deviation from a reference policy π_{ref} , and $r_t(\theta)$ and clip are defined as

$$r_t(\theta) = \frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{\text{old}}}(o_t|q, o_{<t})} \quad (2)$$

$$\text{clip}(x; \epsilon) = \min(1 + \epsilon, \max(1 - \epsilon, x)), \quad (3)$$

respectively. The original implementation of PPO estimated the advantage with Generalized Advantage Estimation (GAE) (Schulman et al., 2015), which uses a value function that is trained jointly with the policy model and a reward model, typically trained in advance with labeled preference data. At a very high level, PPO updates the model by computing how much better or worse a particular action o_t is than average via \hat{A}_t , then updates the model in a constrained region accordingly.

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) was proposed to simplify PPO training, requiring instead a dataset $\mathcal{D} = \{(q, a)\}$ of question q and answer a pairs, modeling an *entire output* o as a single *action*, with original question q being the (*initial*) *state*. GRPO omits the reward model and value function by sampling G complete responses $o^{(1)}, \dots, o^{(G)}$ per question q (known as a *group* of size G) from $\pi_{\theta_{\text{old}}}$, then computing the reward R_i for each $o^{(i)}$ using rule-based

verification with answer a . Then, the advantage $\hat{A}_t^{(i)}$ is computed as $\hat{A}_t^{(i)} = (R_i - \bar{R}) / \sigma_R$, where \bar{R} and σ_R denote the mean and standard deviation of the rewards $\{R_1, \dots, R_G\}$. Above, each token t is assigned the same advantage, and as such, we drop the subscript t moving forward. With this advantage estimator, the GRPO objective is

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{\{(q,a) \sim \mathcal{D}, \{o^{(i)}\}_{i=1}^G \sim \pi_{\theta_{\text{old}}(\cdot|q)}\}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o^{(i)}|} \sum_{t=1}^{|o^{(i)}|} \left(\min \left\{ r_t^{(i)}(\theta) \hat{A}^{(i)}, \right. \right. \right. \\ \left. \left. \left. \text{clip} \left(r_t^{(i)}(\theta); \epsilon \right) \hat{A}^{(i)} \right\} - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) \right) \right], \quad (4)$$

where $r_t^{(i)}(\theta)$ is defined the same as $r_t(\theta)$ in Eq. (2) for output $o^{(i)}$. Many variants of GRPO have recently been proposed, like DAPO (Yu et al., 2025b) (using different values for high/low clipping, among other changes) and DR-GRPO (Liu et al., 2025b) (removing σ_R in the advantage and changing the loss normalization).

3.2 EIS-GRPO: Augmenting with State Equivalence

The vanilla formulation of GRPO considers any two questions q and q' as different states regardless of their specific contents. However, within a real-world collection of inputs, some states may exhibit *equivalence structure* (Li et al., 2006): Different questions may contain the same *relevant* information while differing in *irrelevant* ways (e.g., stylistic). Consider the following toy example:

q = ‘‘Solve [problem]. Think step-by-step.’’

q' = ‘‘Provide reasoning before answering: [problem]’’

Above, q and q' contain the same *relevant* information, i.e., the objective is to solve [problem], which requires the same reasoning process (actions) to solve. However, they differ in *irrelevant* ways (e.g., exact CoT prompt). While identifying equivalent problems in large-scale datasets may be difficult, *transforming a given state (i.e., input) to a set of equivalent versions is easy* via hand-crafted data augmentation strategies (e.g., via paraphrasing).

Rather than sampling G responses from q , we transform q into L equivalent versions, $\mathcal{T}_1(q), \dots, \mathcal{T}_L(q)$, and then sample G/L responses $\{o^{(1,\ell)}, \dots, o^{(G/L,\ell)}\}$ from each $\mathcal{T}_\ell(q)$, for a total of G responses across all $\mathcal{T}_\ell(q)$ ¹. We call the set of responses sampled from a given $\mathcal{T}_\ell(q)$ a *subgroup*. Then, an *advantage that is*

¹In practice, one can vary the number of outputs sampled for each $\mathcal{T}_\ell(q)$. However, for ease of notation and presentation, we assume fixed subgroup sizes of G/L .

computed jointly across all subgroups is added to the subgroup-only advantage, and used to update the *policy conditioned on the transformed state* $\mathcal{T}_\ell(q)$. Concretely, suppose each output $o^{(i,\ell)}$ receives reward $R^{(i,\ell)}$, and denote $R^{[L]} = \{R^{(1,1)}, \dots, R^{(G/L,1)}, \dots, R^{(1,L)}, \dots, R^{(G/L,L)}\}$ as the set of rewards across subgroups and $R^\ell = \{R^{(1,\ell)}, \dots, R^{(G/L,\ell)}\}$ as the set of subgroup rewards. Define $\hat{A}^{(i,\ell)}$ and $r_t^{(i,\ell)}(\theta)$ as

$$\hat{A}^{(i,\ell)} = \frac{(R^{(i,\ell)} - \bar{R}^{[L]})}{\sigma_{R^{[L]}}} + \frac{(R^{(i,\ell)} - \bar{R}^\ell)}{\sigma_{R^\ell}} \quad (5)$$

$$r_t^{(i,\ell)}(\theta) = \frac{\pi_{\theta}(o_t^{(i,\ell)} | \mathcal{T}_\ell(q), o_{<t}^{(i,\ell)})}{\pi_{\theta_{\text{old}}}(o_t^{(i,\ell)} | \mathcal{T}_\ell(q), o_{<t}^{(i,\ell)})}. \quad (6)$$

Then, the EIS-GRPO objective takes the form

$$\mathcal{L}_{\text{EIS-GRPO}}(\theta) = \mathbb{E}_{\{(q,a) \sim \mathcal{D}, \{o^{(i,\ell)}\}_{i=1}^{G/L} \sim \pi_{\theta_{\text{old}}(\cdot|\mathcal{T}_\ell(q))}\}_{\ell=1}^L} \left[\frac{1}{G} \sum_{\ell=1}^L \sum_{i=1}^{G_\ell} \frac{1}{|o^{(i,\ell)}|} \sum_{t=1}^{|o^{(i,\ell)}|} \left(\min \left\{ r_t^{(i,\ell)}(\theta) \hat{A}^{(i,\ell)}, \right. \right. \right. \\ \left. \left. \left. \text{clip} \left(r_t^{(i,\ell)}(\theta); \epsilon \right) \hat{A}^{(i,\ell)} \right\} - \beta D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) \right) \right]. \quad (7)$$

The advantage $\hat{A}^{(i,\ell)}$ comprises two terms: a term computed across subgroups (i.e., a *global* advantage) and a term computed using only subgroup rewards (i.e., a *local* advantage). These two advantages are *complementary*: The global term ties together all subgroups, serving as the mechanism that affirms state equivalence, whereas the local term ensures that each subgroup’s responses are improved independently. That is, the model is taught that all initial states are equivalent and to improve performance for each new initial state. Notably, only using the global advantage may lead to performance degradation, as we show in Sec. 5.2. Two key behaviors arise from the above objective:

- **Inherent output diversity:** The set of outputs across all subgroups is necessarily more diverse than a set sampled from just the original state, which helps in RL training (Liu et al., 2025a).
- **Learned state equivalence:** By comparing outputs from different $\mathcal{T}_\ell(q)$ against each other via $\bar{R}^{[L]}$ and $\sigma_{R^{[L]}}$, each transformed state is modeled as *equivalent* in terms of outcome. Paired with updating the policy with respect to $\mathcal{T}_\ell(q)$, the model learns to be *transformation invariant*.

Some transformations may change the answer a based on \mathcal{T}_ℓ , which we denote as $a^{(\ell)}$. As an example, consider applying EIS-GRPO to train a model to be invariant to multiple choice answer choice labels A, B, C , or D , where each \mathcal{T}_ℓ corresponds to a different multiple choice ordering:

Which animal flies? \mathcal{T}_1 : [A] Cat; [B] Dog; [C] Bird; [D] Pig

\mathcal{T}_2 : [A] Bird; [B] Pig; [C] Dog; [D] Cat

Here, the correct answer *label* must change with each \mathcal{T}_ℓ : $a^{(1)} = [C]$ and $a^{(2)} = [A]$. However, the *substance* of the correct answer (“Bird”) is the same across subgroups, making the subgroups comparable and initial states substantively equivalent.

Connections to existing work. Using data augmentations is inspired by SimCLR (Chen et al., 2020), a contrastive learning framework that maximizes representation similarity between different augmentations (e.g., rotations) of the same image. While SimCLR is explicitly contrastive, the contrastive nature of EIS-GRPO is more implicit through the advantage and policy update. EIS-GRPO can be considered a policy gradient algorithm that leverages *state-equivalence*, connecting it to a line of work in exploiting such structure in Markov Decision Processes (Li et al., 2006; Ravindran and Barto, 2004; Abel et al., 2016; Asadi et al., 2019) and RL (Brunskill and Li, 2013; Mandel et al., 2016). Finally, we note that concurrent work, NoisyRollout (Liu et al., 2025a), in Vision Language Model RL has explored injecting noise into the image prior to generating group outputs. The key difference between NoisyRollout and EIS-GRPO is that they *do not condition on the transformed initial state during the policy update*, i.e., they use $r_t^{(i)}(\theta)$ rather than $r_t^{(i,\ell)}(\theta)$ (Eq. (6)). This difference may appear small but it is crucial, as omitting state-specific conditioning may lead to sub-optimal or incorrect policy updates. For example, when $\mathcal{T}_\ell(q)$ results in a new answer $a^{(\ell)}$, correct behavior is only encouraged when the policy update is with respect to $\mathcal{T}_\ell(q)$. As such, NoisyRollout does not explicitly teach models that augmented initial states are equivalent; rather, as they note, it serves as a group output diversification strategy.

4 Judge Training with EIS-GRPO.

EIS-GRPO, as presented above, is a general-purpose policy-gradient algorithm. Here, we describe its application for judge training. Concretely, our goal is to train a pairwise judge, which takes input a triplet (x, y_1, y_2) of user input x and model responses y_1 and y_2 , parsed into prompt template $T(x, A, B)$, and generates a natural language output $o = \{c, j\}$, where c is a chain-of-thought critique and j is the final judgment (A or B). The prompt template $T(x, A, B)$ includes evaluation and formatting instructions; x is the user input, A is the

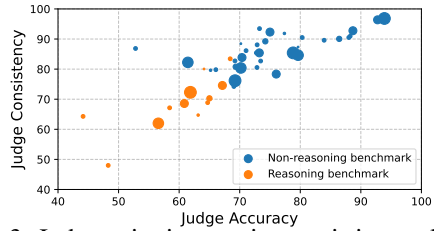


Figure 3: Judge pairwise consistency is inversely correlated with task difficulty, with very poor consistency in reasoning tasks. Plot of 11 judges, with marker size indicating model size across 3 non-reasoning (HHH, LFQA, InstruSum) and 1 reasoning benchmark (JudgeBench).

first response, and B is the *second* response. In what follows, $(A = y_i, B = y_j)$ means that response y_i is presented to the judge first, and response y_j is presented second. *Without loss of generality, we assume that y_1 is the better response.* To train a judge model, we utilize a judgment dataset $\mathcal{D} = \{(x, y_1, y_2)\}$ of pairwise responses.

When judging reasoning, are judges just guessing? Past work (Li et al., 2023; Wang et al., 2023a) has shown that judge models are sensitive to the order of responses. For each pairwise sample, there are two possible orderings: $(A = y_1, B = y_2)$ and $(A = y_2, B = y_1)$. Merely swapping the order can elicit a different chosen response y from the judge. As a result, past work has measured *consistency*, the fraction of pairwise samples where the same response is chosen in both orderings. In Fig. 3, we visualize judge performance on 3 non-reasoning benchmarks (HHH (Askell et al., 2021), LFQA (Xu et al., 2023), InstruSum (Liu et al., 2023)) and a reasoning-focused benchmark, JudgeBench (Tan et al., 2024). The trend is clear: Reasoning evaluation is uniquely difficult, with judges exhibiting behavior more akin to random guessing than substantive assessment, as shown by low consistency.

Judge inconsistency is not a newly discovered phenomenon, with recent judge work taking at proactive measures to mitigate this bias during training, coalescing around using data duplication (Li et al., 2023; Park et al., 2024; Saha et al., 2025) or simply ensuring balanced datasets at larger data scales (Wang et al., 2024a; Cao et al., 2024). The former involves including both $(A = y_1, B = y_2)$ and $(A = y_2, B = y_1)$ orderings in the training set as separate samples, doubling the size of the training set, whereas the latter does not explicitly expose swapped order pairs to the judge to save on training time. Despite targeted training strategies, judges are still not able to consistently assess as task difficulty grows as shown in Fig. 3.

EIS-GRPO for judge training. Data-duplication methods suffer from a key drawback: They do not explicitly link together samples with their position-swapped counterparts; rather, the model consumes each sample independently during training. EIS-GRPO uniquely addresses this drawback by explicitly linking order-swapped training samples, leading to higher consistency *and* better accuracy, as shown in Sec. 5.2. To instill order invariance in the judge, we use 2 subgroups where subgroup 1 orders uses transformation $\mathcal{T}_1(x, y_1, y_2) = \text{T}(x, A = y_1, B = y_2)$ with $a^{(1)} = A$, and subgroup 2 uses transformation $\mathcal{T}_2(x, y_1, y_2) = \text{T}(x, A = y_2, B = y_1)$ with $a^{(2)} = B$. To train J4R, we employ both a judgment reward \mathcal{R}_j and format reward \mathcal{R}_f :

$$\mathcal{R}_j(j, a) = \begin{cases} 1, & j = a \\ 0, & j \neq a \end{cases}$$

$$\mathcal{R}_f(o) = \begin{cases} 0.5, & o \text{ correctly formatted} \\ -0.5, & \text{otherwise} \end{cases}.$$

We compute the final reward $R^{(i,\ell)} = \mathcal{R}_j(o^{(i,\ell)}, a^{(\ell)}) + \mathcal{R}_f(o^{(i,\ell)})$ and use $G = 32$, with 16 for each ordering. For training data, we construct a compact training set of 10K pairwise samples by sampling 20 outputs from Qwen2.5-7B,72B (Yang et al., 2024) and Llama-3.3-70B (Dubey et al., 2024) using the ReClor (Yu et al., 2020) and MATH (Hendrycks et al., 2021) *train* sets and pairing together correct and incorrect responses. We train two judges: J4R-CJ-7B, initialized from CompassJuderger-7B (Cao et al., 2024), and J4R-Qwen-Inst-7B, initialized from Qwen2.5-7B-Instruct. As shown in Sec. 5, EIS-GRPO is effective for both continual and from-scratch judge training. See App. B.1 for hyperparameters and App. E for prompt template.

4.1 ReasoningJudgeBench

Recent judge benchmarks have focused on evaluating robustness to stylistic factors (Liu et al., 2024; Zeng et al., 2023), with reasoning splits using (relatively easy) math and coding problems (e.g., using HumanEvalPack (Muennighoff et al., 2023) prompts). JudgeBench (Tan et al., 2024) takes a step in diversifying reasoning tasks covered, including knowledge-based reasoning (via MMLU-Pro (Wang et al., 2024c)) and more difficult data sources (e.g., LiveBench (White et al., 2024)). The key insight of JudgeBench is that if a strong model struggles to answer a question, that same question will be difficult to assess. Based on this insight, JudgeBench is constructed by: (1)

Sampling multiple outputs from a strong LLM for a question with an objectively correct answer (e.g., math questions) and (2) pairing together a correct and incorrect model response (determined by final output) to form a pairwise test sample.

While JudgeBench is challenging, it contains only 350 pairwise samples that cover only a subset of the myriad of available reasoning tasks. To diversify evaluation, we use the JudgeBench pipeline with GPT-4o to produce ReasoningJudgeBench, which consists of 1,483 response pairs sourced from both established (ARC-Challenge (Clark et al., 2018), ReClor (Yu et al., 2020), StrategyQA (Geva et al., 2021), Folio (Han et al., 2022)) and newer benchmarks (OlympiadBench (He et al., 2024), AIME 2024 and 2025, SuperGPQA (Du et al., 2025), BIG-Bench Extra Hard (Kazemi et al., 2025)). In all, ReasoningJudgeBench consists of 4 splits: Math, multi-hop, domain-specific, and “everyday” reasoning (inductive, causal, and common-sense); See Fig. 1 (center) for a split-level breakdown and App. C for further details.

5 Experimental Results and Analysis

Baselines. We compare J4R against multiple strong, contemporary pairwise judge models: Prometheus-7B, 8x7B (Kim et al., 2024b), CompassJuderger-7B, 14B, 32B (Cao et al., 2024), RISE-Judge-7B, 32B (Yu et al., 2025a), Self-Taught-Evaluator-70B (Wang et al., 2024b), EvalPlanner (Saha et al., 2025), JudgeLRM-7B (Chen et al., 2025a), and RM-R1-7B, 14B, 32B (Chen et al., 2025b). We report official numbers when they exist, or run each judge with its provided prompt template otherwise. We also evaluate o1, o3-mini, GPT-4o, 4o-mini, Qwen2.5-7B (Yang et al., 2024), and DeepSeek-R1 (Guo et al., 2025) on all benchmarks using our prompt (App. E). In App. C.2, we present results for a suite of prompted models on ReasoningJudgeBench.

Benchmarks. To assess judge performance in reasoning domains, we use 3 benchmarks: PPE Best-of- K (Frick et al., 2024) (MBPP, MATH, and GPQA splits), JudgeBench (Tan et al., 2024), and ReasoningJudgeBench. PPE Best-of- K contains $\sim 2,500$ pairwise samples per split, with judges assessing responses sampled from multiple *weaker* generators. In contrast, JudgeBench and ReasoningJudgeBench require judges to assess responses from a stronger model (GPT-4o). For all benchmarks, we adopt a consistent accuracy setup: Each

Model	PPE			JudgeBench				ReasoningJudgeBench				Overall				
	MBPP	MATH	GPQA	Avg.	Knowledge	Math	Reasoning	Code	Avg.	Multi-hop	Math		Everyday	Domain	Avg.	
Small Judges	Prometheus-2-7B	29.70	33.01	27.54	30.09	16.23	33.93	28.57	35.71	24.86	16.56	29.23	17.23	16.33	18.88	24.61
	CompassJudeger-7B	33.89	57.27	40.86	44.02	45.45	73.21	45.92	38.10	49.14	34.95	50.38	32.62	38.00	37.76	43.64
	RISE-Judge-7B	41.18	61.21	36.56	46.35	44.81	57.14	35.71	47.62	44.57	28.60	51.15	31.69	36.00	34.73	41.88
	JudgeLRM-7B [†]	28.80	40.47	24.77	31.37	25.32	44.64	34.69	47.62	33.71	5.52	20.77	14.77	13.67	11.87	25.65
	RM-R1-Distill-7B [†]	31.44	55.04	15.23	33.96	17.53	16.07	19.39	4.76	16.29	12.37	35.38	14.46	3.67	15.10	21.78
	CompassJudeger-14B	36.84	59.57	40.47	45.64	46.75	73.21	45.92	42.86	50.29	37.29	51.54	28.92	36.00	37.69	44.54
	RM-R1-Distill-14B	32.94	72.89	37.46	47.80	42.21	57.14	50.00	42.86	46.86	47.16	68.08	34.77	25.33	43.70	46.12
	J4R-Qwen-Inst-7B (Ours)	29.19	60.43	43.95	43.95	51.30	62.50	52.04	42.86	52.29	38.96	59.62	31.38	46.33	41.74	45.99
	J4R-CJ-7B (Ours)	37.71	64.41	45.82	49.14	55.19	71.43	54.08	56.86	56.86	39.80	61.15	36.92	42.33	45.04	50.35
	Large Judges	RISE-Judge-32B	43.63	69.96	43.93	52.53	46.75	60.71	36.73	52.38	46.86	45.65	55.38	27.69	40.33	42.35
CompassJudeger-32B		41.10	65.16	45.66	50.66	50.00	69.64	53.06	54.76	54.57	49.16	60.77	38.15	38.00	46.53	50.59
RM-R1-Distill-32B		44.10	78.95	40.94	54.71	45.45	64.29	60.20	59.52	54.29	52.51	69.23	33.85	28.00	46.39	51.80
Prometheus-2-8x7B		27.02	37.34	31.01	31.79	37.01	57.14	37.76	21.43	38.57	20.74	38.08	26.46	25.33	25.96	32.11
Self-Taught-70B		41.93	50.08	41.68	44.57	50.65	58.93	42.86	40.48	48.57	45.32	39.62	28.31	35.67	38.64	43.93
EvalPlanner-3.3-70B		-	-	-	-	55.8	69.6	56.1	42.9	56.6	-	-	-	-	-	-
Prompted Judges	Qwen2.5-7B-Instruct	32.54	48.59	34.65	38.61	35.06	44.64	42.86	35.71	38.86	29.60	33.85	29.85	28.67	30.21	35.89
	GPT-4o-mini	39.68	55.86	41.37	45.66	51.30	60.71	44.90	50.00	50.86	51.34	45.77	40.00	36.67	40.39	42.24
	GPT-4o	44.10	61.17	47.23	50.86	50.65	57.14	46.94	47.62	50.29	40.13	54.23	32.31	39.33	45.25	47.18
	o1	52.57	94.92	71.02	72.90	78.57	92.86	82.65	92.86	83.71	77.59	90.38	58.15	65.33	73.10	76.57
	o3-mini	51.08	94.61	66.05	70.64	66.23	85.71	79.59	92.86	76.28	65.72	91.92	49.85	57.33	65.14	70.69
	Deepseek-R1	59.96	93.52	62.07	71.89	71.43	89.29	73.47	71.43	74.86	65.05	86.15	40.00	41.67	58.53	68.43

Table 1: J4R-CJ-7B is the best small judge, almost matching the best large judges despite being 4.5x smaller in size. Best and second-best in each size range are **bolded** and underlined. Avg. is a microaverage across benchmark splits, Overall is an average of the 3 Avg. scores. Highlighting indicates training method: **SFT**, **DPO**, and **RL**. [†] indicates that a judge had systematic issues with output format. All values are *consistent accuracy*.

pairwise sample is evaluated twice, with the order of responses swapped on the second run. Then, a judge is only considered correct if both orders are evaluated correctly. *This setup is the default for JudgeBench, but differs from the original PPE setup, which fixes an ordering; as such, our results for PPE differ from their reported results.*

5.1 Main results

Our main results are presented in Sec. 4.1. J4R-CJ-7B outperforms all small ($\leq 14B$) judge models in aggregate (50.35 vs. the next best 46.12) and roughly matches the performance of the best *large* judge models. In particular, despite being 4.5x smaller in size, J4R-CJ-7B (50.35) almost matches RM-R1-32B (51.80), which was trained with both warm-up SFT with DeepSeek-R1 reasoning traces and GRPO. Such small gap shows that *RL-training alone is insufficient for ensuring robust reasoning evaluation*. As we show later, when given equal compute, J4R-CJ-7B is able to greatly surpass all larger models. J4R-Qwen-Inst-7B (45.99), initialized from a weaker checkpoint, is a strong performing small judge, surpassing GPT-4o-mini (42.24) and nearly matching the twice as large RM-R1-14B (46.12). ReasoningJudgeBench is a significant challenge for judge models, with many capable judges failing to crack 45%. Judges are relatively strong in assessing math, but struggle on the other three splits, showing that evaluating complex *natural language* reasoning is still an open problem.

Training algorithm	JudgeBench		Reas.JudgeBench			
	Acc.	Cons.	Acc.	Cons.		
CompassJudeger-7B (Initial)	49.14	68.86	37.69	68.58		
+SFT	42.86	62.86	32.97	57.11		
+DPO	46.57	68.29	38.84	69.66		
Training data strategy	Norm. data. size	Norm. time	Acc.	Cons.	Acc.	Cons.
CompassJudeger-7B (Initial)	-	-	49.14	68.86	37.69	68.58
+GRPO w/ [Bal.], $G = 32$	1x	1x	52.00	73.43	41.34	75.52
+GRPO w/ [Dup.], $G = 16$	2x	1x	51.14	76.00	38.84	71.61
+GRPO w/ [Dup.], $G = 32$	2x	2x	51.71	77.71	42.01	77.95
+GRPO w/ global adv., $G = 32$	1x	1x	49.14	69.43	37.96	68.58
+EIS-GRPO, $G = 32$ (ours)	1x	1x	56.86	81.14	45.04	80.51

Table 2: Comparison of EIS-GRPO against (top) various common judge training recipes and (bottom) data-driven ways to improve consistency. Both results showcase the benefits of training with EIS-GRPO.

5.2 Additional analysis.

How does initial model impact performance?

As discussed in Sec. 4, we start training from CompassJudeger-7B for J4R-CJ-7B and Qwen2.5-7B for J4R-Qwen-Inst-7B. From Sec. 4.1, we find that starting from a stronger judge checkpoint yields results in better aggregate performance (50.35 vs. 45.99). This result is perhaps not surprising given recent analysis (Yue et al., 2025), which suggests that a primary benefit of RLVR is modifying the model output distribution to favor correct reasoning chains that are already within the initial model’s capabilities. Hence, initializing RL training from a trained judge provides a “higher floor” from which RL can incentivize correct judgments. However, this does not mean EIS-GRPO fails with weaker initial models: J4R-Qwen-Inst-7B improves Qwen2.5-7B from 35.89 aggregate

Baseline Model	Size (M)	Avg. tokens (T_{inf})	δ_{inf}	Baseline Acc.	J4R Acc. (FLOP-matched)
J4R-CJ	7B	270.28	1.0	56.86	-
CompassJudge	14B	213.48	2.0	50.29	57.87 \pm 0.013 (\uparrow 7.58)
CompassJudge	32B	212.87	4.0	54.57	62.00 \pm 0.012 (\uparrow 7.43)
RISE-Judge	32B	532.62	9.0	46.86	64.10 \pm 0.009 (\uparrow 17.24)
RM-R1	14B	1621.83	12.0	46.86	64.48 \pm 0.013 (\uparrow 17.62)
Self-Taught	70B	394.55	15.0	48.57	64.59 \pm 0.007 (\uparrow 16.02)
RM-R1	32B	1792.37	30.0	54.29	64.94 \pm 0.005 (\uparrow 10.65)

Table 3: FLOP-matched evaluation on JudgeBench comparing J4R-CJ-7B to larger judges that may produce longer outputs. With equal compute (last column), J4R-CJ-7B outperforms larger baselines by up to 17.24%.

accuracy to 45.99, a 28% gain, and *outperforms all other 7B judges*, some trained with 20x more data. In all, we show that existing judges optimized for non-reasoning settings were not trained in vain, as they can be continually trained with EIS-GRPO.

How does EIS-GRPO compare with existing judge training recipes? Here, we quantify the advantage EIS-GRPO brings over two common judge training recipes, SFT and DPO. Prior work samples CoT judgments from a teacher model, then uses either SFT (e.g., (Kim et al., 2023; Cao et al., 2024)) or DPO (e.g., (Wang et al., 2024a; Hu et al., 2024)). Using our training set, we follow the approach of (Wang et al., 2024a; Hu et al., 2024; Ye et al., 2024) and distill CoT judgments from Llama-3.1-70B to form both a SFT training set and a paired DPO training set. We ensure each of these training sets are label-balanced, and train for 1 epoch starting from CompassJudge-7B; App. B.3 contains complete details. Sec. 5.2 (top) shows the fundamental limitations of current judge model paradigms: Even using targeted reasoning data, SFT and DPO actually *decrease* performance on JudgeBench relative to the baseline performance. This perhaps surprising result likely stems from two factors: teacher-model responses (1) are *out-of-distribution* and (2) place an upper-bound on performance, as instruct models are relatively weak reasoning evaluators (See App. C.2).

How does EIS-GRPO improve over other data-driven consistency strategies? As discussed in Sec. 4, past work has used label-balanced training (Wang et al., 2024a; Cao et al., 2024) or data duplication (Li et al., 2023; Park et al., 2024; Saha et al., 2025) to mitigate inconsistency. Here, we show that EIS-GRPO improves upon both of these approaches. From our train set, we form a data balanced version ([Bal]), where we assign 50% of our dataset to have label A and 50% label B, and a duplication variant ([Dup]), which doubles the size

of our training set by including each ordering of responses. For [Bal], we run GRPO with the same training setup as J4R-CJ-7B (Sec. 4). For [Dup], we consider two setups. The first uses $G = 16$ and doubles the rollout batch size, resulting in the same number of model updates and training steps as J4R. The second applies the exact same training setup as J4R, resulting in twice the training steps.

We report normalized training time with respect to J4R-CJ-7B, along with judge performance, in Sec. 5.2 (bottom). Both [Bal] and [Dup] result in some gains in accuracy and consistency, but fall significantly short of EIS-GRPO. For the same number of model updates, EIS-GRPO improves JudgeBench accuracy 9.4% and 11.2% over [Bal] and [Dup], respectively. Further, EIS-GRPO results in more positionally robust judges, as shown by significantly higher consistency values. We also ablate using only the global advantage (i.e., only the first term of $\hat{A}^{(i,\ell)}$ in Eq. (6)), and find that its performance lags all other strategies, highlighting the necessity of both global and local information. We present a deeper exploration in App. D.1.

Given equal inference-time compute (FLOPs), how does J4R match up with larger judges?

Here, we present a FLOP-matched evaluation on JudgeBench using the common FLOP approximation $2MT_{\text{inf}}$ (Snell et al., 2024; Sardana et al., 2023), where M is the model size and T_{inf} is the number of inference tokens. For J4R-CJ-7B and all judges of size $\geq 14B$, we compute the average number of output tokens per sample on JudgeBench, allowing us to compute a FLOP-scaling ratio $\delta_{\text{inf}} = \frac{M_b T_{b,\text{inf}}}{M_{\text{J4R}} T_{\text{J4R},\text{inf}}}$, where subscript b denotes baseline and J4R denotes J4R-CJ-7B. We then sample δ_{inf} parallel responses from J4R-CJ-7B with temperature 1.0 and aggregate via majority vote (i.e., self-consistency (Wang et al., 2022)). To reduce compute time, we sample a fixed set of 128 responses from the J4R-CJ-7B and randomly subsample for each δ_{inf} . We repeat each subsampling 100 times and report mean and standard deviation. Sec. 5.2 shows the benefit of matched compute: the performance of J4R-CJ-7B improves up to 14%. Given the same compute as RM-R1-32B, J4R outperforms RM-R1-32B by 10.65 absolute percent.

How does reasoning-specific training affect other domains? We measure the difference between J4R and initial models on LFQA (Xu et al., 2023) (long-form QA), HHH (Askell et al., 2021) (safety and helpfulness), and InstruSum (Liu et al.,

Model	HHH	LFQA	InstruSum
CompassJudge-7B	83.26	70.00	64.23
J4R-CJ-7B	77.83	67.69	66.91
Qwen2.5-7B-Instruct	72.40	50.77	51.82
J4R-Qwen-Inst-7B	76.47	62.69	60.83

Table 4: Group size G ablation.

G	16	24	32	64
Acc.	40.93	40.66	45.04	44.71
Cons.	74.92	72.49	80.51	79.97

Table 5: Reasoning-specific training improves weaker models, but degrades chat eval. for stronger models.

2023) (summarization quality). Our results, shown in Sec. 5.2, are in-line with expectations: Specializing a strong model for a specific domain reduces general abilities, as shown with degradations in chat-focused settings (HHH and LFQA), but improves performance when the task is reasoning-demanding, like contextual comprehension. However, when the initial model is weak, “low-hanging fruit” gains are available, and reasoning specific training holistically improves performance.

How does group size affect EIS-GRPO? In Table 4, we use ReasoningJudgeBench to quantify the effect of group size G . We find that performance increases as G increases, with the largest gain occurring from 24 and 32. This highlights the importance of using sufficient G to ensure strong contrastive signals between subgroups. Using a larger group size of 64 does not yield further gains.

6 Conclusion

We demonstrate the effectiveness of EIS-GRPO for training positionally robust judges, improving evaluation in reasoning settings. We diversify existing judge benchmarks with ReasoningJudgeBench, which reveals that judges have significant room for growth in evaluating in reasoning settings.

Limitations

Our work demonstrates the effectiveness of EIS-GRPO for training positionally robust judge models. As our work is focused on quantifying the performance of our methodology (analyzing model initialization, quantifying advantages over other training approaches, etc.), our complete training pipeline has under-optimized components. For example, we did not optimize our training dataset, opting for a simple approach for generating training pairs used in past work (Saha et al., 2025). We additionally did not employ recent developments building on GRPO, such as those presented

in DAPO (Yu et al., 2025b). We note, however, that EIS-GRPO is easily adaptable to such developments.

We additionally chose to focus on the pairwise evaluation setting, as it is a common setting for judges. As such, our work does not explore how to exploit state-equivalence in settings such as single rating evaluation or process reward modeling. We consider finding modeling state-equivalence in these settings an exciting line of future work.

Finally, we note that EIS-GRPO is naturally impacted by choice of group size G , as shown in our last experiment. EIS-GRPO takes inspiration from contrastive learning frameworks enforce similarity and dissimilarity within *batches* of samples (Chen et al., 2020; He et al., 2020; Lee et al., 2019); The group size G in EIS-GRPO can be thought of as analogous to contrastive learning batch size. A common criticism of such contrastive learning approaches is that they require large training batch sizes (Gao et al., 2021; Chen et al., 2022), necessitating large amounts of GPU VRAM. While EIS-GRPO requires relatively large G , a key difference between EIS-GRPO and contrastive learning approaches is that the *minimum required GPU VRAM does not increase as G increases*. This is because policy updates occur sequentially. Complications with gradient accumulation for contrastive learning approaches, described in (Gao et al., 2021), are absent in EIS-GRPO. That is, sampling higher G does not increased hardware minimums. However, we acknowledge that sampling sufficiently large G may result in relatively longer training times.

References

- David Abel, David Hershkowitz, and Michael Littman. 2016. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, pages 2915–2923. PMLR.
- Afra Feyza Akyürek, Ekin Akyürek, Aman Madaan, Ashwin Kalyan, Peter Clark, Derry Wijaya, and Niket Tandon. 2023. R14f: Generating natural language feedback with reinforcement learning for repairing model outputs. *arXiv preprint arXiv:2305.08844*.
- Mahsa Asadi, Mohammad Sadegh Talebi, Hippolyte Bourel, and Odalric-Ambrym Maillard. 2019. Model-based reinforcement learning exploiting state-action equivalence. In *Asian Conference on Machine Learning*, pages 204–219. PMLR.
- Amanda Askill, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, and 1 others.

2021. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*.
- Emma Brunskill and Lihong Li. 2013. Sample complexity of multi-task reinforcement learning. *arXiv preprint arXiv:1309.6821*.
- Maosong Cao, Alexander Lam, Haodong Duan, Hongwei Liu, Songyang Zhang, and Kai Chen. 2024. Compassjudge-1: All-in-one judge model helps model evaluation and evolution. *arXiv preprint arXiv:2410.16256*.
- Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, and 1 others. 2025. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*.
- Chi-Min Chan, Chunpu Xu, Jiaming Ji, Zhen Ye, Pengcheng Wen, Chunyang Jiang, Yaodong Yang, Wei Xue, Sirui Han, and Yike Guo. 2025. J1: Exploring simple test-time scaling for llm-as-a-judge. *arXiv preprint arXiv:2505.11875*.
- Changyou Chen, Jianyi Zhang, Yi Xu, Liqun Chen, Jiali Duan, Yiran Chen, Son Tran, Belinda Zeng, and Trishul Chilimbi. 2022. Why do we need large batchsizes in contrastive learning? a gradient-bias perspective. *Advances in Neural Information Processing Systems*, 35:33860–33875.
- Nuo Chen, Zhiyuan Hu, Qingyun Zou, Jiaying Wu, Qian Wang, Bryan Hooi, and Bingsheng He. 2025a. Judgelrm: Large reasoning models as a judge. *arXiv preprint arXiv:2504.00050*.
- Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR.
- Xiuxi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. 2025b. **Rm-r1: Reward modeling as reasoning**. *Preprint*, arXiv:2505.02387.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, and 1 others. 2025. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Benjamin Feuer, Micah Goldblum, Teresa Datta, Sanjana Nambiar, Raz Besaleli, Samuel Dooley, Max Cembalest, and John P Dickerson. 2024. Style outweighs substance: Failure modes of llm judges in alignment benchmarking. *arXiv preprint arXiv:2409.15268*.
- Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios N Angelopoulos, Jiantao Jiao, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. 2024. How to evaluate reward models for rlhf. *arXiv preprint arXiv:2410.14872*.
- Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021. Scaling deep contrastive learning batch size under memory limited setup. *arXiv preprint arXiv:2101.06983*.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.
- Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, and 1 others. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhen-ting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, and 1 others. 2022. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.
- Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- Ilgee Hong, Changlong Yu, Liang Qiu, Weixiang Yan, Zhenghao Xu, Haoming Jiang, Qingru Zhang, Qin

- Lu, Xin Liu, Chao Zhang, and 1 others. 2025. Thinkrm: Enabling long-horizon reasoning in generative reward models. *arXiv preprint arXiv:2505.16265*.
- Xinyu Hu, Li Lin, Mingqi Gao, Xunjian Yin, and Xiaojun Wan. 2024. Themis: A reference-free nlg evaluation language model with flexibility and interpretability. *arXiv preprint arXiv:2406.18365*.
- Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, and 1 others. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.
- Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Peter Chen, and 1 others. 2025. Big-bench extra hard. *arXiv preprint arXiv:2502.19187*.
- Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, Peifeng Wang, Silvio Savarese, and 1 others. 2025. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *arXiv preprint arXiv:2504.09037*.
- Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process reward models that think. *arXiv preprint arXiv:2504.16828*.
- Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. 2023. Causal reasoning and large language models: Opening a new frontier for causality. *arXiv preprint arXiv:2305.00050*.
- Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoon Yun, Seongjin Shin, Sungdong Kim, James Thorne, and 1 others. 2023. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*.
- Seungone Kim, Juyoung Suk, Ji Yong Cho, Shayne Longpre, Chaeun Kim, Dongkeun Yoon, Guijin Son, Yejin Cho, Sheikh Shafayat, Jinheon Baek, and 1 others. 2024a. The biggest bench: A principled benchmark for fine-grained evaluation of language models with language models. *arXiv preprint arXiv:2406.05761*.
- Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024b. Prometheus 2: An open source language model specialized in evaluating other language models. *arXiv preprint arXiv:2405.01535*.
- Wouter Kool, Herke van Hoof, and Max Welling. 2019. Buy 4 reinforce samples, get a baseline for free!
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.
- Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300*.
- Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. 2023. Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470*.
- Lihong Li, Thomas J Walsh, and Michael L Littman. 2006. Towards a unified theory of state abstraction for mdps. *AI&M*, 1(2):3.
- Xiangyan Liu, Jinjie Ni, Zijian Wu, Chao Du, Longxu Dou, Haonan Wang, Tianyu Pang, and Michael Qizhe Shieh. 2025a. Noisyrollout: Reinforcing visual reasoning with data augmentation. *arXiv preprint arXiv:2504.13055*.
- Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. 2024. Rm-bench: Benchmarking reward models of language models with subtlety and style. *arXiv preprint arXiv:2410.16184*.
- Yixin Liu, Alexander R Fabbri, Jiawen Chen, Yilun Zhao, Simeng Han, Shafiq Joty, Pengfei Liu, Dragomir Radev, Chien-Sheng Wu, and Arman Cohen. 2023. Benchmarking generation and evaluation capabilities of large language models for instruction controllable summarization. *arXiv preprint arXiv:2311.09184*.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025b. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.
- Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. 2025c. Inference-time scaling for generalist reward modeling. *arXiv preprint arXiv:2504.02495*.

- Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. 2024. Generative reward models. *arXiv preprint arXiv:2410.12832*.
- Travis Mandel, Yun-En Liu, Emma Brunskill, and Zoran Popovic. 2016. Efficient bayesian clustering for reinforcement learning. In *IJCAI*, pages 1830–1838.
- Nat McAleese, Rai Michael Pokorny, Juan Felipe Ceron Uribe, Evgenia Nitishinskaya, Maja Trebacz, and Jan Leike. 2024. Llm critics help catch llm bugs. *arXiv preprint arXiv:2407.00215*.
- Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro von Werra, and Shayne Longpre. 2023. Octopack: Instruction tuning code large language models. *arXiv preprint arXiv:2308.07124*.
- Allen Nie, Yuhui Zhang, Atharva Shailesh Amdekar, Chris Piech, Tatsunori B Hashimoto, and Tobias Gerstenberg. 2024. Moca: Measuring human-language model alignment on causal and moral judgment tasks. *Advances in Neural Information Processing Systems*, 36.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.
- Junsoo Park, Seungyeon Jwa, Meiyong Ren, Daeyoung Kim, and Sanghyuk Choi. 2024. Offsetbias: Leveraging debiased data for tuning evaluators. *arXiv preprint arXiv:2407.06551*.
- Balaraman Ravindran and Andrew G Barto. 2004. Approximate homomorphisms: A framework for non-exact minimization in markov decision processes.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.
- Swarnadeep Saha, Xian Li, Marjan Ghazvininejad, Jason Weston, and Tianlu Wang. 2025. Learning to plan & reason for evaluation with thinking-llm-as-a-judge. *arXiv preprint arXiv:2501.18099*.
- Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. 2023. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. *arXiv preprint arXiv:2401.00448*.
- John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Nihar B Shah, Sivaraman Balakrishnan, Joseph Bradley, Abhay Parekh, Kannan Ramch, Martin J Wainwright, and 1 others. 2016. Estimation from pairwise comparisons: Sharp minimax bounds with topology dependence. *Journal of Machine Learning Research*, 17(58):1–47.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseek-math: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and 1 others. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.
- Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. 2024. Judgebench: A benchmark for evaluating llm-based judges. *arXiv preprint arXiv:2410.12784*.
- Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Galouédec. [TRL: Transformer Reinforcement Learning](#).
- Tu Vu, Kalpesh Krishna, Salaheddin Alzubi, Chris Tar, Manaal Faruqui, and Yun-Hsuan Sung. 2024. Foundational autoraters: Taming large language models for better automatic evaluation. *arXiv preprint arXiv:2407.10817*.
- Peifeng Wang, Austin Xu, Yilun Zhou, Caiming Xiong, and Shafiq Joty. 2024a. Direct judgement preference optimization. *arXiv preprint arXiv:2409.14664*.
- Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023a. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.

- Tianlu Wang, Ilia Kulikov, Olga Golovneva, Ping Yu, Weizhe Yuan, Jane Dwivedi-Yu, Richard Yuanzhe Pang, Maryam Fazel-Zarandi, Jason Weston, and Xian Li. 2024b. Self-taught evaluators. *arXiv preprint arXiv:2408.02666*.
- Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O’Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023b. Shepherd: A critic for language model generation. *arXiv preprint arXiv:2308.04592*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, and 1 others. 2024c. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddhartha Naidu, and 1 others. 2024. Livebench: A challenging, contamination-free llm benchmark. *arXiv preprint arXiv:2406.19314*.
- Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. 2025. J1: Incentivizing thinking in llm-as-a-judge via reinforcement learning. *arXiv preprint arXiv:2505.10320*.
- Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Do, Wenyu Zhan, and 1 others. 2024. Enhancing llm reasoning via critique models with test-time and training-time supervision. *arXiv preprint arXiv:2411.16579*.
- Zhihui Xie, Liyu Chen, Weichao Mao, Jingjing Xu, Lingpeng Kong, and 1 others. 2025. Teaching language models to critique via reinforcement learning. *arXiv preprint arXiv:2502.03492*.
- Fangyuan Xu, Yixiao Song, Mohit Iyyer, and Eunsol Choi. 2023. A critical evaluation of evaluations for long-form question answering. *arXiv preprint arXiv:2305.18201*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, and 1 others. 2023. Retroformer: Retrospective large language agents with policy gradient optimization. *arXiv preprint arXiv:2308.02151*.
- Ziyi Ye, Xiangsheng Li, Qiuchi Li, Qingyao Ai, Yujia Zhou, Wei Shen, Dong Yan, and Yiqun Liu. 2024. Beyond scalar reward model: Learning generative judge from preference data. *arXiv preprint arXiv:2410.03742*.
- Jiachen Yu, Shaoning Sun, Xiaohui Hu, Jiayu Yan, Kaidong Yu, and Xuelong Li. 2025a. Improve llm-as-a-judge ability as a general ability. *arXiv preprint arXiv:2502.11689*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, and 1 others. 2025b. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.
- Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. *arXiv preprint arXiv:2002.04326*.
- Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*.
- Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2023. Evaluating large language models at evaluating instruction following. *arXiv preprint arXiv:2310.07641*.
- Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*.
- Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. 2025. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future. *arXiv preprint arXiv:2504.12328*.
- Yilun Zhou, Austin Xu, Peifeng Wang, Caiming Xiong, and Shafiq Joty. 2025. Evaluating judges as evaluators: The jets benchmark of llm-as-judges as test-time scaling evaluators. *arXiv preprint arXiv:2504.15253*.

A Extended Related Work and Discussion

A.1 Related work: Pointwise generative verification.

While pairwise evaluation forms the focus of our work, evaluation can also be done in a pointwise manner. Pointwise generative verifiers are another subset of the generative judge paradigm that assess one response at a time. Earlier work (Kim et al., 2024b) explored using per-instance references and rubrics to stabilize evaluation, while more recent

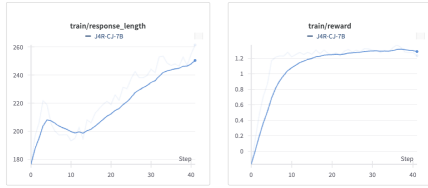


Figure 4: Reward and length dynamics of training J4R-CJ-7B.

work (Hu et al., 2024) explores larger-scaled training with preference optimization. Pointwise evaluators do not necessarily need to be standalone models: Many pairwise judges (e.g., (Wang et al., 2024a; Vu et al., 2024; Hu et al., 2024; Kim et al., 2024b; Li et al., 2023)) are also capable pointwise evaluators, assigning scores of 1-5 or binary pass/fail scores based on certain evaluation criteria. Past work (Park et al., 2024) has also found that pointwise training can improve pairwise performance. Concurrently, long CoT distillation has been used to improve such models (Khalifa et al., 2025) for math evaluation. Closely related are critic models, which are models trained at providing feedback to models. Such models are typically trained with human critiques (Wang et al., 2023b), distillation (Xi et al., 2024) or more recently, RL (McAleese et al., 2024; Akyürek et al., 2023; Yao et al., 2023; Xie et al., 2025).

The efficient pointwise nature makes such evaluators attractive choices in settings like inference-time verification. However, recent work has highlighted the difficult nature of single-instance verification in reasoning intensive domains (Zhou et al., 2025). The pairwise paradigm offers stability in assessment, as relative comparisons a valuable “other item” to anchor assessment against. Broadly, relative (ordinal) assessment has been shown to be easier for humans than pointwise (cardinal) assessment (Shah et al., 2016), a fact that also appears to carry over to generative verifiers.

B Model training details

B.1 EIS-GRPO training details

To train our judge to be strong in reasoning domains, we create a reasoning-based training set of pairwise model responses. In particular, we sample responses from both weaker (Qwen2.5-7B) and stronger models (Qwen2.5-72B and Llama3.3-70B) to ensure diversity. We choose two datasets with training sets, MATH (Hendrycks et al., 2021) and ReClor (Yu et al., 2020) with objectively cor-

rect answers. We then prompt each model to answer each question in the training set after outputting a CoT, sampling 20 responses using temperature of 1.0. Based on final outputs, we form pairs of “correct” and “incorrect” responses. This can be viewed as a form of synthetic data generation, as employed by prior work, e.g., (Wang et al., 2024b; Saha et al., 2025). All training was conducted on an 8xH200 GPU node. We use rollout batch size of 256, train batch size of 128, constant learning rate of $1e - 6$, clip $\epsilon = 0.2$, and the KL $\beta = 0.0001$.

B.2 EIS-GRPO training dynamics

We visualize training dynamics Fig. 4, we present the judge response length and average reward dynamics during training. Response length grows steadily as training progresses, with the final judge response length nearly 50% longer than the starting point. Reward grows steadily over the first quarter of training before transitioning to a more gradual increase. We additionally find that JudgeBench and ReasoningJudgeBench performance, both accuracy and consistency, both improve relatively steadily over the course of training.

B.3 DPO and SFT judge baseline details

From our train set, we distill 20 CoT judgments from Llama-3.1-70B-Instruct per prompt using a temperature of 0.7 and divide the 20 teacher outputs into correct and incorrect outputs based on ground-truth label. We then form DPO pairs by selecting one correct and incorrect response. We ensure that the dataset has balanced labels. For samples where the teacher model produces the same judgment for all 20 samples (i.e., all correct or all incorrect), we include the sample as a non-CoT training sample: We remove the requirement for CoT judgment in the prompt, asking the judge to respond directly with "A" or "B" without CoT. This type of data sample was found to help judge training (Wang et al., 2024a) and allows us to use all of our training data, for fair comparison. We additionally add an SFT loss during DPO, as is commonly done in judge training (Wang et al., 2024a; Ye et al., 2024). For DPO, we use $\beta = 0.1$. For SFT, we use the correct samples from the DPO train set. We train both SFT and DPO models for 1 epoch using TRL (von Werra et al.) with commonly found hyperparameters (e.g., (Li et al., 2023; Ivison et al., 2023)). We use an effective batch size of 64 and utilize AdamW and a cosine learning rate scheduler

with peak learning rates of $1e-5$ (SFT) and $5e-7$ (DPO).

C ReasoningJudgeBench: Details and additional results.

Constructing ReasoningJudgeBench follows a three stage approach: (1) We select a challenging benchmark with objectively correct outcomes, (2) We prompt a strong LLM (GPT-4o) to provide 20 CoT responses for each input, (3) We construct pairs using outcome verification, keeping one correct and one incorrect response per prompt. We sample prompts where the model gets all 20 responses correct or incorrect. Because judges are known to exhibit length bias (Zeng et al., 2023; Park et al., 2024), when constructing our pairs, we keep the pair of positive and negative responses with *minimal difference in length* in order to create more challenging pairs.

We chose both established and recent benchmarks as our source datasets. This allows us to balance challenging, newer samples with existing samples that are fundamentally challenging, i.e., are still difficult to answer for models despite being publicly released for some time. In particular, we use

- **ARC-Challenge** (Clark et al., 2018): An established multiple-choice QA benchmark.
- **ReClor** (Yu et al., 2020) (Validation set): A multiple-choice QA benchmark focused on reading comprehension, sourced from professional standardized tests.
- **StrategyQA** (Geva et al., 2021): An implicit, multi-hop reasoning QA benchmark.
- **Folio** (Han et al., 2022) (Validation set): A human-written first-order logic benchmark
- **AIME 2024 and 2025**: Competition-level math problems
- **OlympiadBench** (He et al., 2024) (Text-only): Olympiad-level math problems.
- **SuperGPQA** (Du et al., 2025) (Random subsample): An expanded version of GPQA (Rein et al., 2024), scaled to 285 disciplines.
- **BIG-Bench Extra Hard** (Kazemi et al., 2025) (Boolean Expressions, Causal Understanding (Nie et al., 2024; Kiciman et al., 2023), Disambiguation QA, Hyperbaton, Multi-step arithmetic): A harder variant of BigBench-Hard (Suzgun et al., 2022).

The above selection of benchmarks was chosen to fill the gaps of evaluation. Namely,

JudgeBench (Tan et al., 2024) draws from a small number of benchmarks, limiting the types of reasoning covered. ReasoningJudgeBench represents an expansion of reasoning domains evaluated, including the myriad of unique reasoning tasks from BIG-Bench Extra Hard. Due to the limited size and difficulty of AIME 2024 and 2025, GPT-4o was only able to generate 15 valid pairs. As a result, we utilized GPT-4.1-mini to augment math subset. Both models’ responses are included in the Math subset. The per-split data source breakdown is as follows, where we abbreviate BIG-Bench Extra Hard as BBEH.

- **Multi-hop reasoning** [598 samples]: ARC-Challenge (60), BBEH Boolean Expressions (161), Folio (88), ReClor (289)
- **Math** [260 samples]: AIME 2024 and 2025 (38), BBEH Multi-step arithmetic (38), Olympiad-Bench (184).
- **Domain** [300 samples]: SuperGPQA (300)
- **Everyday** [325]: BBEH Causal Understanding (89), BBEH Disambiguation QA (68), BBEH Hyperbaton (114), StrategyQA (54).

C.1 Model inference details.

All models of size 72B or smaller were run locally on vLLM (Kwon et al., 2023) on nodes of 1,4, or 8xA100s. OpenAI model inference was run using their Batch API with the following endpoints: `gpt-4o-2024-08-06`, `gpt-4o-mini-2025-04-16`, `o3-mini-2025-01-31`, `o1-2024-12-17`, and `o4-mini-2025-04-16`. All reasoning models are evaluated with “reasoning effort” set to medium (default). DeepSeek-R1 and V3 and Llama-3.1-405B were accessed through together.ai.

C.2 Additional evaluation results

Here, we evaluate a suite of instruct and reasoning models on ReasoningJudgeBench. For judges that allow ties, we follow (Tan et al., 2024) and consider a run with a tie to be correct if the non-tie response is correct. We present results in App. C.2. We find that reasoning models tend to perform well on evaluation, with top performers being `o1` at 73.10% and `o4-mini` at 69.39%. The top models are also the most consistent, most exceeding 80%. Among instruction-tuned judges, GPT-4.1 performs well, passing the performance of DeepSeek-R1.

	Model	Multi-hop	Math	Everyday	Domain	Avg.	Consistency
Small Judges	Prometheus-2-7B	16.56	29.23	17.23	16.33	18.88	47.61
	CompassJudge-7B	34.95	50.38	32.62	38.00	37.76	67.16
	RISE-Judge-7B	28.60	51.15	31.69	36.00	34.73	66.82
	JudgeLRM-7B [†]	5.52	20.77	14.77	13.67	11.87	54.28
	RM-R1-Distill-7B [†]	12.37	35.38	14.46	3.67	15.10	48.11
	CompassJudge-14B	37.29	51.54	28.92	36.00	37.69	63.79
	RM-R1-Distill-14B	47.16	68.08	34.77	25.33	43.70	65.54
	J4R-Qwen-Inst-7B (Ours)	41.97	59.62	31.38	36.33	41.60	73.57
	J4R-CJ-7B (Ours)	43.81	61.15	36.92	42.33	45.04	80.98
Large Judges	RISE-Judge-32B	45.65	55.38	27.69	40.33	42.35	73.23
	CompassJudge-32B	49.16	60.77	38.15	38.00	46.53	74.04
	RM-R1-Distill-32B	52.51	69.23	33.85	28.00	46.39	66.35
	Prometheus-2-8x7B	20.74	38.08	26.46	25.33	25.96	54.89
	Self-Taught-70B	45.32	39.62	28.31	35.67	38.64	63.32
Prompted Inst. Judges	Qwen2.5-7B-Instruct	29.60	33.85	29.85	28.67	30.21	57.79
	Qwen2.5-14B-Instruct	41.30	39.23	31.69	25.67	35.67	62.58
	Qwen2.5-32B-Instruct	48.49	38.46	35.69	40.33	42.28	68.64
	Qwen2.5-72B-Instruct	50.33	48.46	32.31	39.00	43.76	72.76
	Llama-3.1-8B-Instruct	25.75	33.85	20.00	17.67	24.28	48.89
	Llama-3.1-70B-Instruct	48.66	38.08	28.92	30.67	38.84	67.16
	Llama-3.3-70B-Instruct	51.51	45.00	28.31	10.00	36.88	72.15
	Llama-3.1-405B-Instruct	57.53	55.00	32.00	36.67	47.27	73.50
	GPT-4o-mini	40.13	45.77	40.00	36.67	40.39	70.47
	GPT-4o	51.34	54.23	32.31	39.33	45.25	73.50
	GPT-4.1-nano	26.25	42.31	32.00	27.00	30.48	50.51
	GPT-4.1-mini	59.03	77.69	42.15	49.00	56.57	80.31
	GPT-4.1	68.39	76.54	42.15	46.67	59.68	81.73
	Deepseek-V3	55.52	58.85	36.92	42.00	49.29	74.51
Prompted Reas. Judges	o1	77.59	90.38	58.15	65.33	73.10	87.53
	o3-mini	65.72	91.92	49.85	57.33	65.14	83.41
	o4-mini	74.92	89.62	52.31	59.33	69.39	82.87
	Deepseek-R1	65.05	86.15	40.00	41.67	58.53	72.89

Table 6: Complete evaluation results for ReasoningJudgeBench, covering small and large judges (presented in main body), prompted instruction tuned models, and prompted reasoning models. † denotes systematic issues with output formatting.

D Additional analysis and qualitative examples

D.1 Why does only using a global advantage fail?

In Sec. 5.2, we showed that running EIS-GRPO *without* individual advantages resulted in dramatically different results. Concretely, if one retains the subgroup sampling but calculates the advantage as $\tilde{A}^{(i,\ell)}$, where

$$\tilde{A}^{(i,\ell)} = (R^{(i,\ell)} - \bar{R}^{[L]}) / \sigma_{R^{[L]}}, \quad (8)$$

the performance drops significantly (e.g., from 56.86 on JudgeBench to 49.14). Here we provide

an illustrative example into the drawbacks of only using the global advantage.

Suppose we have two subgroups of 16 responses, denoted subgroup A and subgroup B, where subgroup A’s performance is very strong (12 responses of reward 1.5 and 4 responses of reward 1.0) and subgroup B’s performance is very poor (12 responses of reward 0.0 and 4 responses of reward 1.0). Now, consider the three candidates for advantage computation: (1) Separate groups (vanilla GRPO), (2) Global advantage only, and (3) Global and Local advantage (EIS-GRPO).

Separate groups. Within subgroup A, the reward 1.5 responses will receive advantage 0.577, while the reward 1.0 responses receive advantage -1.732. For subgroup B, the reward 1.0 responses will receive advantage 1.732, while the reward 0.0 responses receive advantage -0.577. Each subgroup behaves as expected: Better than average rewards are encouraged, and worse than average rewards are discouraged.

Global advantage only. Computing a mean and standard deviation across both subgroups yields the following. Within subgroup A, the reward 1.5 responses will receive advantage 1.044, while the reward 1.0 responses receive advantage 0.285. For subgroup B, the reward 1.0 responses will receive advantage 1.732, while the reward 0.0 responses receive advantage -1.234. At first glance, such advantage assignment is intuitive: The reward 1.5 responses get encouraged heavily, while the reward 0.0 responses get heavily discouraged.

However, the key drawback is that the poor responses in subgroup B do not get the necessary positive reinforcement. While a reward 1.0 may be roughly average on a global scale, they are exceptional by subgroup B standards. However, these responses are not getting as much positive advantage because other, stronger responses exist in *a different subgroup*.

Global and Local advantage (EIS-GRPO). By summing both global and local advantages, we arrive at the following advantage assignment. Within subgroup A, the reward 1.5 responses will receive advantage 1.621, while the reward 1.0 responses receive advantage -1.447. For subgroup B, the reward 1.0 responses will receive advantage 2.017, while the reward 0.0 responses receive advantage -1.811. The sum of the two advantages fixes the issues with Global only: Now, reward 1.0 responses within subgroup B are properly assigned advantage that results in improvement, while the global advantage ties together responses across subgroups, promoting state equivalence.

D.2 Qualitative J4R-CJ-7B output examples

Example 1: From ReasoningJudgeBench, source dataset AIME 2024.

Question =====

Solve the following math problem step by step. The last line of your response should be of the form Answer: \$Answer (without quotes) where \$Answer is the answer to the problem.

Find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b .

Remember to put your answer on its own line after "Answer:".

Response A (Incorrect) =====

To solve this problem, we need to understand what the numbers 17_b and 97_b mean in terms of the base b and how 17_b might be a divisor of 97_b .

Converting Numbers from Base b

- Convert 17_b :
 - In base b , 17_b means $(1 \cdot b + 7)$.
 - So, $17_b = b + 7$.
- Convert 97_b :
 - In base b , 97_b means $(9 \cdot b + 7)$.
 - So, $97_b = 9b + 7$.

Divisibility Condition

We require 17_b to be a divisor of 97_b . Thus:

$97_b = k \cdot 17_b$
for some integer k . Substituting the values we found:

$$9b + 7 = k(b + 7)$$

Expanding the right side:

$$9b + 7 = kb + 7k$$

Solving the Equation

Rearrange the equation:

$$\begin{aligned} 9b + 7 &= kb + 7k \\ 9b + 7 &= kb + 7k \\ 9b - kb &= 7k - 7 \end{aligned}$$

Factor out b on the left:

$$b(9 - k) = 7(k - 1)$$

Determine Appropriate Values

Since we need $b > 9$, try small integer values for k that could make b an integer greater than 9.

$$b = \frac{7(k-1)}{9-k}$$

Test Values of k :

- If $k = 2$: $b = \frac{7(2-1)}{9-2} = \frac{7}{7} = 1$ (Not valid since $b > 9$)
- If $k = 3$: $b = \frac{7(3-1)}{9-3} = \frac{14}{6} = 2.\overline{3}$ (Not valid)
- If $k = 4$: $b = \frac{7(4-1)}{9-4} = \frac{21}{5} = 4.2$ (Not valid)
- If $k = 5$: $b = \frac{7(5-1)}{9-5} = \frac{28}{4} = 7$ (Not valid)
- If $k = 6$: $b = \frac{7(6-1)}{9-6} = \frac{35}{3} = 11.\overline{6}$ (Not valid as an integer)
- If $k = 7$: $b = \frac{7(7-1)}{9-7} = \frac{42}{2} = 21$

A valid integer b is 21. Thus, $b = 21$ is a base greater than 9 for which 17_b divides 97_b exactly.

Conclusion

Having checked for divisibility circumstances and calculations, we conclude that the only base $b > 9$ that satisfies the condition is:

$$b = 21$$

Therefore, the sum of all such bases is:
 $\text{\text{Answer: } 21}$

Response B (Correct) =====

To solve this problem, we first need to convert the numbers (17_b) and (97_b) from base (b) to their decimal equivalents:

1. **Convert (17_b) into decimal:** The number (17_b) represents $(1 \times b + 7)$. Therefore, in decimal, $(17_b = b + 7)$.

2. **Convert (97_b) into decimal:** The number (97_b) represents $(9 \times b + 7)$. Therefore, in decimal, $(97_b = 9b + 7)$.

The problem states that (17_b) (which is $(b + 7)$) is a divisor of (97_b) (which is $(9b + 7)$).

This implies that:

$9b + 7 = k(b + 7)$
for some integer (k) . Expanding the right-hand side gives:
 $9b + 7 = kb + 7k$

Subtract (kb) from both sides:

$9b + 7 - kb = 7k$

This simplifies to:

$(9 - k)b + 7 = 7k$

Rearrange to solve for (b) :

$(9 - k)b = 7k - 7$

$b = \frac{7(k - 1)}{9 - k}$

Since (b) must be an integer greater than 9, we find values of (k) such that $(9 - k)$ divides $7(k - 1)$ and the resulting (b) is an integer greater than 9.

Test small values of (k) :

1. $(k = 1)$: $b = \frac{7(1-1)}{9-1} = \frac{0}{8} = 0$, $\text{\text{(not valid since } b > 9)}$

2. $(k = 2)$: $b = \frac{7(2-1)}{9-2} = \frac{7}{7} = 1$, $\text{\text{(not valid since } b > 9)}$

3. $(k = 3)$: $b = \frac{7(3-1)}{9-3} = \frac{14}{6}$, $\text{\text{(not an integer)}}$

4. $(k = 4)$: $b = \frac{7(4-1)}{9-4} = \frac{21}{5}$, $\text{\text{(not an integer)}}$

5. $(k = 5)$: $b = \frac{7(5-1)}{9-5} = \frac{28}{4} = 7$, $\text{\text{(not valid since } b > 9)}$

6. $(k = 6)$: $b = \frac{7(6-1)}{9-6} = \frac{35}{3}$, $\text{\text{(not an integer)}}$

7. $(k = 7)$: $b = \frac{7(7-1)}{9-7} = \frac{42}{2} = 21$, $\text{\text{(valid)}}$

8. $(k = 8)$: $b = \frac{7(8-1)}{9-8} = \frac{49}{1} = 49$, $\text{\text{(valid)}}$

9. $(k = 9)$: $b = \frac{7(9-1)}{9-9}$ is undefined because this involves division by zero.

The valid integer bases (b) for which $(b > 9)$ are $(b = 21)$ and $(b = 49)$. Therefore, their sum is: $21 + 49 = 70$

Answer: 70

Judge explanation =====

Explanation: Both Assistant A and Assistant B provided detailed solutions to the problem. However, there is a critical mistake in the final step of Assistant A's response. After correctly identifying that $(b = 21)$

(b) is a valid base, Assistant A incorrectly concludes that this is the only valid base. In contrast, Assistant B correctly identifies both valid bases $(b = 21)$ and $(b = 49)$ and calculates their sum correctly.

Assistant B's response is more accurate because it correctly identifies all valid bases that satisfy the given conditions and calculates their sum correctly. Assistant A's response, while detailed, misses the second valid base $(b = 49)$ and therefore provides an incorrect final answer.

Verdict: $[B > A]$

Example 2: From ReasoningJudgeBench, source dataset BIGBench Extra Hard, Disambiguation QA split

Question =====

Answer the following multiple choice question. Think step-by-step before outputting your answer. When you output your answer, repeat your answer choice 5 times. For example, if you select A, output your answer as AAAAA.

Here is a sentence with pronoun(s) whose antecedent(s) can either be derived from the context or is ambiguous. Sentence: Two friends are travelling in Europe during the summer. One of them proposed to go to London to visit his uncle who lives there. He has not seen members of his family for a while.

Which of the following options correctly explains the antecedent(s) of the pronoun(s)?

- (A) The uncle has not seen members of his family for a while.
- (B) One of the two friends has not seen members of his family for a while.
- (C) It is ambiguous who has not seen members of his family.
- (D) One of the two friends is a female.
- (E) None of the above.

Output your answer strictly following this format:

Reasoning: Your reasoning here

Answer: Your choice, repeated 5 times

Response A (Correct) =====

Reasoning: Let's analyze the sentence step by step. The sentence describes a situation involving two friends traveling in Europe. One of them proposed going to London to visit "his" uncle, which suggests that the uncle belongs to one of the two friends. The next part of the sentence states, "He has not seen members of his family for a while." The antecedent of "He" could logically be the friend who proposed the visit, implying that this friend has not seen members of his family for a while. However, it could also be interpreted that the uncle has not seen his family for a while, making the antecedent ambiguous. Therefore, without additional context, it is unclear who "he" refers to in this scenario, as both interpretations are plausible. Hence, (C) is the correct choice, indicating that it is ambiguous who has not seen members of his family.

Answer: CCCCC

Response B (Correct) =====

Reasoning: In the given sentence, the pronoun "he" is

used in "He has not seen members of his family for a while." To determine the antecedent of "he," we consider the context provided. The sentence starts with "Two friends are travelling...", and later says "One of them proposed to go to London to visit his uncle who lives there." Here, "his uncle" suggests the uncle of one of the friends. We are trying to figure out who "he" refers to in the last part of the sentence. Since two male pronouns ("his" and "he") are used, it's reasonable to think both times it refers to the same person, meaning it refers to the same friend who suggested visiting his uncle. Thus, "he" most likely refers to one of the two friends. Therefore, the pronoun "he" in the final sentence does not introduce ambiguity within the context, and it refers to one of the two friends.

Answer: BBBB

Judge explanation =====

Explanation: Both Assistant A and Assistant B provided thoughtful analyses of the sentence to determine the antecedent of the pronoun "he." However, their conclusions differ. Assistant A correctly identifies that the antecedent of "he" is ambiguous because it could refer to either the friend who proposed the visit or the uncle. Assistant B, on the other hand, concludes that "he" refers to one of the two friends, which is a valid interpretation but not the only one, as the sentence does not provide enough context to definitively determine who "he" is. Therefore, Assistant A's answer is more accurate because it acknowledges the ambiguity in the sentence, which is the correct response given the options provided.

Verdict: [[A>B]]

E Prompt templates

E.1 J4R prompt templates.

We present the prompt template used to evaluate J4R-CJ-7B, J4R-Qwen-Inst-7B, and CompassJuder below. Originally, CompassJuder evaluates on JudgeBench using the ArenaHard prompt, which first tasks the judge with *solving the original problem* prior to evaluation. This prompt muddies the waters between *generation ability* and *evaluation ability*. As our goal is to precisely analyze the degree to which our judges can *evaluate*, and the degree to which EIS-GRPO improves evaluation capabilities, we disentangle the two abilities by prompting the judge to only evaluate.

Using our prompt, improves the 7B model performance, but leads to slight degradations for the 14B and 32B versions, as noted in the main text. Furthermore, we are unable to reproduce officially reported CompassJuder numbers, the evaluation of which was run by JudgeBench. For example, JudgeBench reports that CompassJuder-7B achieves 46.00%. Using JudgeBench's own scripts, we obtain a score of 34.43%, whereas with our own implementation, we obtain a score of 45.71%. As

a result, we report deviations with respect to our reproduced numbers: 45.71% for 7B, 54.29% for 14B, and 59.14% for 32B.

```
### System Prompt
Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user prompt displayed below. You will be given assistant A's answer and assistant B's answer. Your job is to evaluate which assistant's answer is better.
```

When evaluating the assistants' answers, identify any mistakes or inaccurate information. Focus on the content each response, and select the response that arrives at the correct response and with the most sound and logical reasoning. Avoid any biases, such as order of responses, length, or stylistic elements like formatting.

Before providing your final verdict, think through the judging process and output your thoughts as an explanation

After providing your explanation, you must output only one of the following choices as your final verdict with a label:

1. Assistant A is better: [[A>B]]
2. Assistant B is better: [[B>A]]

Use the following template:
Explanation: Your detailed thought process as an explanation.
Verdict: [[A>B]] or [[B>A]].

```
### User Prompt
<|User Prompt|>
{question}
```

```
<|The Start of Assistant A's Answer|>
{response_a}
<|The End of Assistant A's Answer|>
```

```
<|The Start of Assistant B's Answer|>
{response_b}
<|The End of Assistant B's Answer|>
```

E.2 Benchmark generation prompts

Below, we provide two sample prompts for generating model responses to (1) multiple choice questions and (2) math questions when constructing ReasoningJudgeBench.

```
### Multiple choice prompt
```

Answer the following multiple choice question. Think step-by-step before outputting your answer. When you output your answer, repeat your answer choice 5 times. For example, if you select A, output your answer as AAAAA.

```
{optional context}
Question: {question}
```

```
{choices}
```

Output your answer strictly following this format:
Reasoning: Your reasoning here
Answer: Your choice, repeated 5 times

Math prompt

Solve the following math problem step by step. The last line of your response should be of the form Answer: \$Answer (without quotes) where \$Answer is the answer to the problem.

{question}

Remember to put your answer on its own line after "Answer:".