On the Shelf Life of Fine-Tuned LLM-Judges: Future-Proofing, Backward-Compatibility, and Question Generalization

Anonymous authors
Paper under double-blind review

Abstract

The LLM-as-a-judge paradigm is widely used in both evaluating free-text model responses and reward modeling for model alignment and finetuning. Recently, finetuning judges with judge-specific data has emerged as an often preferred choice over directly prompting frontier models as judges, as the former achieves better performance with smaller model sizes while being more robust to common biases. However, the standard evaluation ignores several practical concerns of finetuned judges regarding their real world deployment. In this paper, we identify and formalize three aspects that affect the *shelf life* of these judges: *future-proofing* and *backward-compatibility* – how well judges finetuned on responses by today's generator models perform on responses by future models or past models, as well as *question generalization* – how well judges generalize to unseen questions at test time. We study these three aspects under a unified framework with varying train and test distributions in two reasoning datasets, three SFT- and DPO-based fine-tuning algorithms, and three different backbone models. Experiments suggest that future-proofing is challenging for most models, while backward-compatibility is relatively easy, with DPO-trained models consistently *improving* performance. We further find that continual learning provides a more balanced adaptation to shifts between older and newer response distributions than training solely on stronger or weaker responses. Moreover, all models observe certain degrees of performance degradation when moving from questions seen during training to unseen ones, showing that current judges do not fully generalize to unseen questions. These findings provide insights into practical considerations for developing and deploying judge models in the face of ever-changing generators.

1 Introduction

Automatic evaluators have become a central part of the large language model (LLM) development cycle. They serve both as reward models during training (Stiennon et al., 2020; Ouyang et al., 2022; Yuan et al., 2024) and as verifiers in inference-time compute scaling (Zhou et al., 2025; Kim et al., 2025; Singhi et al., 2025). In the LLM-as-judge paradigm, a generative language model evaluates the outputs of other models for a given input question, providing a scalable approach to automatic evaluation. Past work on LLM-as-judges began with zero-shot prompting of capable LLMs (Liu et al., 2023; Dubois et al., 2023). However, such judges have been shown to be prone to various biases, such as stylistic bias (Zeng et al., 2024; Raina et al., 2024), length bias (Zheng et al., 2023; Zeng et al., 2024), and positional bias (Wang et al., 2023; Pezeshkpour & Hruschka, 2024). As a result, recent efforts have finetuned specialized evaluators Li et al. (2024b); Kim et al. (2024a); Vu et al. (2024), which have been shown to be more robust to common forms of bias (Zhu et al., 2025; Wang et al., 2024a; Park et al., 2024a) while matching the performance of larger prompted models.

Although recent advances in judge model finetuning have largely focused on developing training methodology Chen et al. (2025a;c), little attention has been devoted to understanding how these models behave as a function of their training inputs. In this work, we investigate this gap by asking three key questions: First, can judge models trained on fixed datasets of input questions, model responses, and ground-truth verdicts accurately evaluate the responses of newer models, i.e., are judges *future-proof*? Second, if we train a judge on up-to-date responses from newer models, can it

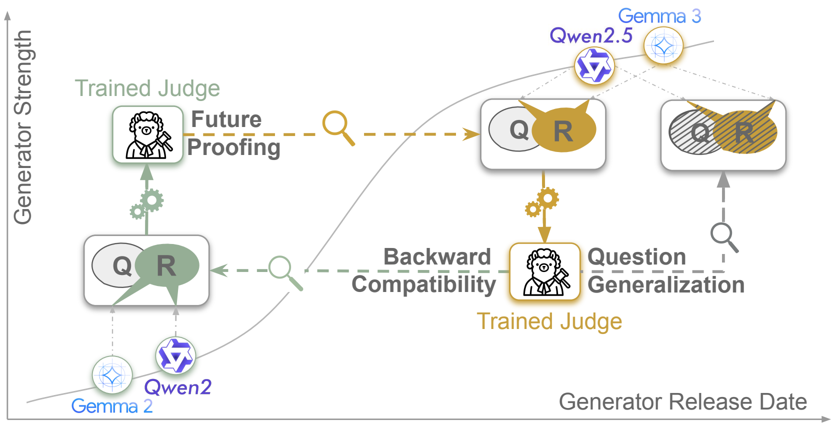


Figure 1: High-level overview of our setup for studying *Future-Proofing*, *Backward-Compatibility*, and *Question Generalization* through the lens of generalization and robustness to input distribution shifts. Q and R represent questions and responses, respectively, with responses generated by the shown generator models (Gemma2, Qwen2, Gemma3, Qwen2.5). *Future-Proofing* evaluates how well judges trained on responses from weaker, older generators (green: Gemma2, Qwen2) assess responses from stronger, newer generators (yellow: Gemma3, Qwen2.5). *Backward-Compatibility* examines the reverse direction. *Question Generalization* measures performance on in-distribution questions and corresponding responses that were both not included (dashed Q and R) in the training.

reliably evaluate responses from older models, i.e., is the trained judge *backward-compatible*? Third, fixing the response generating models, how reliably can judges assess questions that differ from those seen during training, i.e., do they *generalize to new questions*? We examine these questions, as illustrated in Figure 1, through the lens of generalization and robustness, aiming to understand the *shelf life* of trained judges.

In this work, we propose a *dual-distribution* formulation of automatic evaluation. Concretely, we model the judge's input as comprising elements drawn from two distinct distributions: the *question distribution*, which characterizes the input questions to be evaluated, and the *response distribution*, which characterizes the responses to be judged. We study the performance of trained judges when responses are drawn from relatively weak and strong generators, henceforth referred to as weak responses and strong responses. We also examine how well trained judges evaluate questions that are (1) seen during training but paired with new responses, and (2) completely unseen during training. By focusing on weak and strong generators and novel questions, we gain insights into the shelf life of trained judge models through four practical questions:

- **Future-proofing.** Given a judge trained on responses from older ("weak") models, how accurately can it evaluate responses from newer ("strong") models? If the goal is to evaluate strong responses, how much benefit do practitioners gain by training on strong responses rather than weak ones?
- **Backward-compatibility.** Given a judge trained on responses from newer ("strong") models, can it reliably assess responses from older ("weak") models? If the goal is to evaluate weak responses, does training a judge on strong responses provide any benefit?
- **Continual learning.** Compared to judges trained only on weak or strong responses, how well does a continually trained judge adapt to distribution shifts between the two response distributions?
- **Question generalization.** Does judge performance depend on whether a question was seen during training? Even for seen questions, can a judge reliably assess new responses?

Using two verifiable datasets (DeepScaleR and MMLU-Pro), we set up a suite of controlled experiments to analyze the shelf life of judge models, training across three backbone models of varying sizes and capabilities and three popular judge-training recipes. Our findings reveal that fine-tuned judges struggle to evaluate newer, stronger model responses and therefore require training with up-to-date response distribution. Once trained on newer, stronger responses, judges exhibit some degree of backward-compatibility. Continual training provides a more balanced adaptation to shifts between older and newer response distributions than training solely on stronger or solely on weaker responses. Finally, we find that fine-tuned judges struggle to generalize to new questions. In all, our findings inform the development and deployment of future generations of finetuned judge models.

2 Background and Related Work

2.1 An overview of finetuned judges.

LLM-based judges are automatic evaluators that evaluate LLM outputs given some evaluation criteria. While many judges accommodate different evaluation tasks, such as single rating ("Rate this response on a scale of 1-5") (Hu et al., 2024) or classification ("Is this response appropriate?") (Vu et al., 2024), the dominant evaluation paradigm LLM-based judges are deployed with is *pairwise evaluation.* Here, a judge is given a question and two candidate responses, and tasked with selecting the "better" response according to some criteria. Formally, the judge performs the transformation

$$(Q, R_1, R_2) \quad \longrightarrow \quad (C, \hat{V}), \quad C \text{ optional}, \tag{1}$$

where Q is the question, R_1, R_2 are the two candidate responses, C is an optional chain-of-thought explanation, and $\hat{V}$ is the verdict of which response is better. We denote $x = (Q, R_1, R_2) \sim \mathcal{X}$ to be the judge input and $y = (C, \hat{V})$ to be the judge output. Pairwise judges are typically evaluated using accuracy or consistent accuracy, the latter accounting for response-order bias as detailed in Appendix D. Due to its popularity and practicality, pairwise evaluation forms the focus of our study.

Past work in judge finetuning uses supervised finetuning (SFT) (Li et al., 2024b; Kim et al., 2024b; Zhu et al., 2025), preference optimization methods, like direct preference optimization (DPO) (Wang et al., 2024a; Ye et al., 2024; Saad-Falcon et al., 2024), or more recently, reinforcement learning with verifiable rewards (RLVR) (Chen et al., 2025a;c; Whitehouse et al., 2025; Xu et al., 2025b). Starting from a dataset of $(x, V^\star(x))$ pairs, where $V^\star$ denotes the ground-truth verdict/label, each approach constructs training samples differently: SFT and DPO approaches sample judge outputs from a *teacher model*, then use $V^\star(x)$ to categorize judge outputs as either correct outputs y^+ or incorrect outputs y^-. Then, the judge is trained on (x, y^+) pairs for SFT and (x, y^+, y^-) triplets for DPO. On the other hand, RL approaches directly make use of the $(x, V^\star(x))$ pairs, omitting the need for teacher model explanations.

2.2 Related work

Distribution Shifts and Generalization. Distribution shift, the mismatch between training and evaluation data, is a long-standing challenge in machine learning (Hendrycks & Dietterich, 2019; Koh et al., 2021). Early computer vision studies demonstrated significant accuracy drops under minor perturbations (Hendrycks & Dietterich, 2019), and WILDS extended this to real-world domain shifts (Koh et al., 2021). In LLMs, the problem is amplified as both data and model capabilities evolve over time (Shi et al., 2025). Recent frameworks explore how models transfer across distributions. *Easy-to-hard generalization* examines whether training on easier tasks transfers to harder ones (Sun et al., 2024), which relates to scalable oversight where only easy tasks can be reliably supervised (Amodei et al., 2016); task-difficulty can be estimated using either model or data-centric measures (Swayamdipta et al., 2020). *Weak-to-strong generalization* investigates improving strong models using supervision derived from weaker ones (Burns et al., 2023). Our setting complements these efforts by focusing on distribution shifts that arise from an *evolving population of generators* and by evaluating how judge models adapt to both weak-to-strong and strong-to-weak shifts.

Analyzing LLM-as-Judge. Prior work analyzes systematic judge biases such as positional (Wang et al., 2023; Li et al., 2024b), length (Zeng et al., 2024; Park et al., 2024b), and self-preference (Panickssery et al., 2024; Chen et al., 2025b). Prompt design, instructions, and scoring format strongly affect reliability (Li et al., 2024a), with pairwise judgments often reducing noise and aligning better with human preferences than pointwise scores (Tripathi et al., 2025; Jeong et al., 2024). Other works have emphasized the importance of carefully selecting reference answers (Krumdick et al., 2025), linking to how generator capabilities influence the judge's inputs (Tan et al., 2025). While most studies consider *static* judges on *fixed* datasets, we instead analyze judges in a dynamic setting where generators change over time, introducing response-distribution shifts that motivate our metrics for *future-proofing*, *backward-compatibility*, and *question generalization*.

3 Automatic Evaluation as a Dual-Distribution Problem

We propose a novel formulation of the automatic evaluation problem in terms of two distributions: the question distribution and the response distribution. Concretely, let $\mathcal{Q}$ denote the distribution of questions Q, and let $\mathcal{R}$ denote the distribution of responses R. For pairwise judges, the input distribution $\mathcal{X}$ therefore takes the form

$$\mathcal{X} = \mathcal{Q} \times \mathcal{R} \times \mathcal{R} \tag{2}$$

The question distribution is defined by characteristics such as semantic content (e.g., domains like medical, legal, finance, scientific, or math) and question difficulty (e.g., difficulty can be defined by pedagogical levels, such as high school vs. olympiad-level math problems). For example, we can consider all questions in GSM8K (Cobbe et al., 2021) to come from the same question distribution, as they are all arguably of similar difficulty and semantic content. The response distribution defines the characteristics of the model responses being evaluated, such as style, capability-specific content, or model-family-specific quirks. We denote the training and test input distributions to be

$$\mathcal{X}^{train} = \mathcal{Q}^{train} \times \mathcal{R}^{train} \times \mathcal{R}^{train} \quad \text{and} \quad \mathcal{X}^{test} = \mathcal{Q}^{test} \times \mathcal{R}^{test} \times \mathcal{R}^{test} \tag{3}$$

respectively. Notably, the two responses come from the same generating model, as described in the data construction details in Section 4.2. Separating the *question distribution* $\mathcal{Q}$ from the *response distribution* $\mathcal{R}$ reflects two real-world sources of shift: (1) the emergence of more capable generators (an evolving $\mathcal{R}$), and (2) the introduction of new questions (an evolving $\mathcal{Q}$). This decomposition allows us to isolate and quantify the impact of each factor on judge performance. In Section 5, we instantiate this framework using the weak response distribution $\mathcal{R}_{weak}$ and the strong response distribution $\mathcal{R}_{strong}$ to simulate a model-development timeline (older, weaker vs. newer, stronger responses and LLMs), along with question splits Q drawn from $\mathcal{Q}$ that are either seen or unseen during training. Informally, weak (strong) responses are drawn from LLMs with relatively low (high) accuracy on questions Q; we precisely describe generator strength in Section 4. This instantiation enables us to investigate the four practical questions mentioned in Section 1 regarding the *shelf life* of judges. The specifics of how dual-distribution formalization supports our analysis are detailed in Section 5, with a concise connection provided in Appendix E.

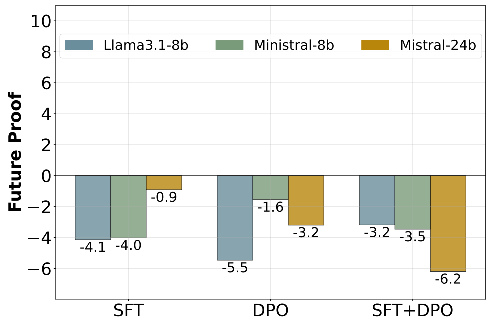
4 Experimental Setup

4.1 Gauging Generator Strength.

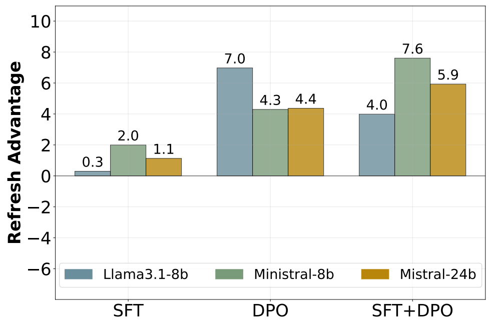
We ground our study in two datasets with verifiable solutions: DeepScaleR (Luo et al., 2025) and MMLU-Pro (Wang et al., 2024c). DeepScaleR contains 40K Olympiad-style, reasoning-intensive math problems with gold answers. MMLU-Pro, by contrast, provides verifiable MCQ-style, knowledge-intensive questions spanning 14 diverse domains, including STEM, humanities, social sciences, law, business, psychology, and philosophy, enabling us to study judge shelf-life across a broad range of domains. For generators, we utilise a diverse set of popular instruction-tuned models, which are listed in Table 1. For each generator, we sample 20 responses per question and measure its strength using Pass@1. Pass@1 captures the probability that a uniformly sampled attempt is correct and yields two clearly separated clusters, as shown in Figures 7 and 8 in Appendix B, where recent or larger models achieve substantially higher scores than smaller or old. Based on this distinguishable performance difference, we cluster low- and high-performing generators into weak (Gemma-2-9B, Qwen-2-7B, Llama-3.1-8B, Ministral-8B) and strong (Gemma-3-12B, Qwen-2.5-7B, Qwen-2.5-32B, Llama-3.3-70B, Mistral-Small-24B) groups, respectively, and use these clusters to define our response-distribution shifts. Further details on generator selection and strength estimation are provided in Appendix B.

4.2 Training Setup.

Dataset Construction. To create the training and evaluation splits, we first construct pairwise input samples for the judge, following prior work (Tan et al., 2025; Wang et al., 2024b). For each question, we sample multiple responses from each generator, and each response is then labeled as "correct" or "incorrect" according to the ground-truth answer $A^{\star}$. We then form response pairs, where each pair consists of one correct response and one incorrect response, resulting in a pairwise



(a) `FutureProof.`



(b) `RefreshAdvantage.`

Figure 2: *Future-proofing of DeepScaler-Trained Judges.* (a) Future-proofing measured by `FutureProof`; negative values show degraded performance on stronger responses. All models and recipes performance degrade, indicating poor evaluation of newer, stronger responses. (b) Benefits of re-training on strong responses, measured by `RefreshAdvantage`. Re-training consistently improves performance, with the largest gains under DPO.

sample with an objectively correct answer. Importantly, responses in a pair are drawn from a single generator only. Based on the generator strengths defined above, we construct datasets of aggregated pairwise samples consisting exclusively of either weak or strong responses, which we refer to as our *weak dataset* and *strong dataset*, respectively.

Judge Data Distillation & Training Objectives. We train judges using three commonly adopted recipes: supervised fine-tuning (SFT) (Li et al., 2024b; Kim et al., 2024a; Vu et al., 2024), direct preference optimization (DPO) (Hu et al., 2024; Wang et al., 2024b), and a combined SFT and DPO objective (Wang et al., 2024a; Ye et al., 2024; Saad-Falcon et al., 2024). As these recipes require supervision, specifically, the CoT explanation C (Sec. 2), we adopt the common *teacher model* convention (Li et al., 2024b; Wang et al., 2024a). Based on the ground-truth verdict V^*, we categorize responses as correct (positive) samples y^+ or incorrect (negative) samples y^-. Positive samples are then used for SFT, whereas positive-negative pairs are used for DPO-based recipes.

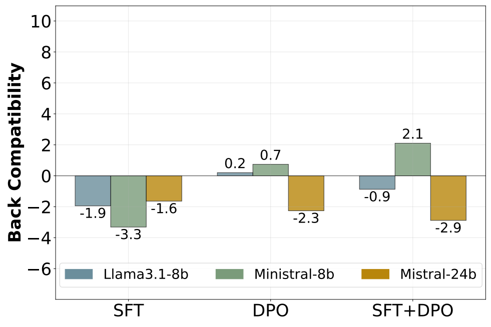
Training and Evaluation Splits. To analyze the four practical questions described in Section 1 using the dual-distribution framework from Section 3, we split the weak and strong datasets into training and test sets. For testing, we construct two distinct splits: an *unseen-questions* split and a *seen-questions* split. The unseen-questions split contains questions not present during training, while seen-questions split reuses training questions but samples *new* responses, with pairs constructed following the same process as described above. Unless otherwise specified, we use the unseen-questions split for evaluation. We choose three models to train: Llama-3.1-8B, Ministral-8B, and Mistral-24B, covering a range of model sizes and intrinsic strengths.

We provide more details on different aspects of the training setup in Appendix C.

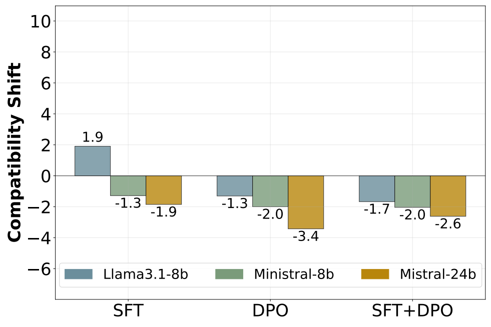
5 Experimental Results

In this section, we present our analysis setup and findings on future-proofing, backward-compatibility, and question-generalization of judge models. Our analysis builds on the dual-distribution framework introduced in Section 3, where judge inputs are factorized into a question distribution $\mathcal{Q}$ and a response distribution $\mathcal{R}$. We instantiate the response distribution at two levels of generator strength: $\mathcal{R}_{weak}$ (older, less capable models) and $\mathcal{R}_{strong}$ (newer, more capable models). The question distribution $\mathcal{Q}$ remains fixed but varies in whether a question was seen or unseen during training. In this way, our setup simulates model development timelines. We measure judge performance using consistent accuracy, as defined in Appendix D. Raw consistent accuracy scores are reported in Table 4 of Appendix D, and serve as the foundation for the results below.

Notation. For clarity, we denote the consistent accuracy of a judge J_t trained on response distribution t as $\mathrm{Acc}_e(J_t)$, where $t \in \{\text{weak}, \text{strong}\}$. The subscript e indicates the evaluation distribution,



(a) `BackCompatibility`.



(b) `CompatibilityShift`.

Figure 3: *Backward-Compatibility of DeepScaler-Trained Judges.* (a) `BackCompatibility` of judges trained on strong responses when evaluating older responses; positive values indicate improved performance relative to older-judge baselines. Judges trained on newer responses show good `BackCompatibility`, with minimal drops—or even absolute *gains*. (b) Despite strong absolute performance, newer judges still face a distribution shift, reflected by `CompatibilityShift`, with performance drops relative to evaluating strong responses. (c) Compared with future-proofing metrics in Figure 2, backward-compatibility metrics are smaller, indicating that strong-response–trained judges are more backward-compatible than weak-response–trained judges are future-proof.

with $e \in \{\text{weak}, \text{strong}\}$. Thus, $\text{Acc}_e(J_t)$ ties back to our dual-distribution formalism: it measures the accuracy of a judge trained on distribution t when evaluated on responses from distribution e.

5.1 How future-proof are judge models?

Experimental Setup. To study *future-proofing* in our simulated model development timeline, we design the following setup: weak generators serve as proxies for existing LLMs, and judges are trained on their responses. Strong generators represent newly released LLMs with greater capabilities. By future-proofing, we refer to how well weak-response-trained judges can evaluate responses from newer, stronger LLMs. Specifically, we quantify future-proofing using the following metrics:

`FutureProof` is defined as the difference in the performance of a weak-response-trained judge between the weak and strong evaluation sets:

$$\texttt{FutureProof} = \text{Acc}_{strong}(J_{weak}) - \text{Acc}_{weak}(J_{weak}). \tag{4}$$

This measures the change in performance when the evaluation distribution shifts from $\mathcal{R}^{test}_{weak}$ to $\mathcal{R}^{test}_{strong}$, i.e., a *weak-to-strong* response distribution shift. A positive value indicates relatively better performance on strong responses, while a negative value indicates degradation; thus, higher values correspond to more future-proof judges.

`RefreshAdvantage` is defined as the gain from re-training judges with strong responses:

$$\texttt{RefreshAdvantage} = \text{Acc}_{strong}(J_{strong}) - \text{Acc}_{strong}(J_{weak}). \tag{5}$$

This can be viewed as the *data advantage* from changing the training response distribution from $\mathcal{R}^{train}_{weak}$ to $\mathcal{R}^{train}_{strong}$ when evaluating on $\mathcal{R}^{test}_{strong}$. Higher values indicate greater benefit from re-training judges with the latest and stronger responses.

`FutureProof` Findings: For all models and training recipes, we plot the `FutureProof` values on DeepScaleR in Figure 2a. Across all settings, we do not observe any instance where judges generalize to new or stronger responses, with all `FutureProof` values being negative. Interestingly, no discernible trend emerges across training recipes or model families. Generally, we find that SFT leads to higher degradations in smaller models, but a smaller degradation in the large judge. In all, our results show that current judge training approaches do not produce judges capable of reliably generalizing to new, more capable model responses. Beyond lack of generalization, current judge recipes do not exhibit consistent trends across models or scales. These findings align with those on MMLU-Pro, which we discuss in more detail in Appendix F.1. In the absence of recipe-specific or model-specific trends, we recommend evaluating `FutureProof` on a model-by-model basis.

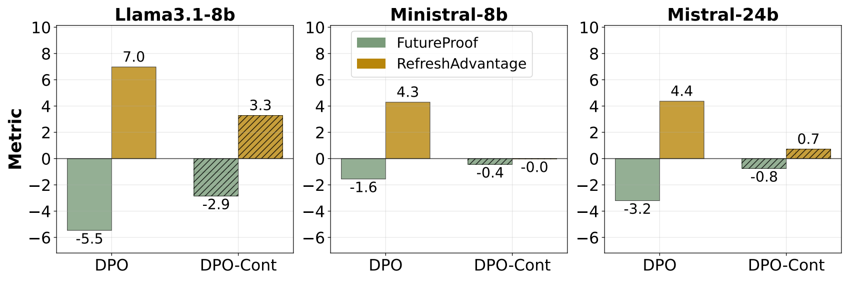


Figure 4: Changes in future-proofing metrics when replacing a weak-response-trained judge (solid) with a continually trained judge (dashed). We observe a decrease in `RefreshAdvantage` and an increase in `FutureProof`, with values approaching zero for a couple of models. This suggests that continual training enables judges to evaluate strong responses more effectively than weak-trained judges, as well as strong-trained judges, and adapts better to the weak-to-strong response shift.

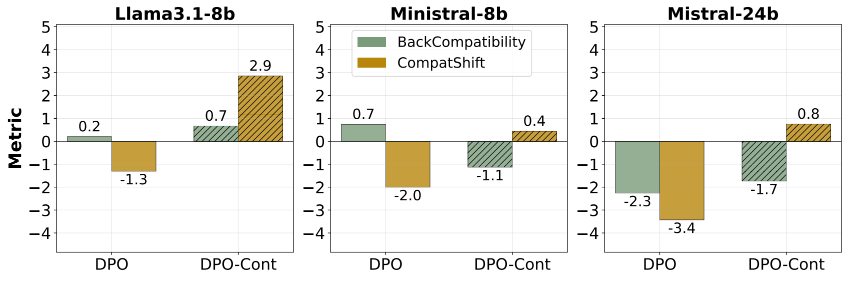


Figure 5: Changes in backward-compatibility metrics when replacing a strong-response-trained judge (solid) with a continually-trained judge (dashed). We see an increase in `BackCompat` for a couple of models, suggesting that continual training can help models better evaluate weak responses than purely strong-trained judges. We also observe an increase in `CompatShift`, showing that continually trained judges adapt better to the strong-to-weak response shift.

`RefreshAdvantage` Findings. Our results on DeepScaleR, presented in Figure 2b, indicate that re-training with up-to-date responses consistently leads to performance gains. In particular, across all training recipes and backbone models, we observe positive `RefreshAdvantage` values. Training recipes also follow a clear trend: retraining with SFT yields minimal but positive gains, whereas DPO yields the largest improvements, providing up to 7.6 absolute percentage points for larger models. The SFT+DPO loss provides additional benefit over DPO alone for couple of models. We further observe that as judge model size increases, updating training data has a larger impact for DPO-based approaches. For example, with DPO, Mistral-24B exhibits an absolute gain of 7.6 percentage points compared to its 8B counterpart, Mistral-8B, which improves by 4.3 points. These trends are consistent with corresponding findings on MMLU-Pro, discussed further in Appendix F.1. Overall, we conclude that evaluating the most capable models requires training judges on their outputs; relying on stale training data leaves substantial performance gains unrealized.

5.2 How backward-compatible are judge models?

Experimental setup. Now, we extend our setup for *future-proofing* in Section 5.1 to study *backward-compatibility* in a simulated model development timeline. A judge trained on strong or newer generator responses represents the current judge, which is adept at evaluating new responses, while weak generators represent older LLMs with lower capabilities. By *backward-compatibility*, we refer to how well strong-response-trained judges can evaluate the responses of older, weaker generators. Specifically, we quantify backward-compatibility using the following metrics:

`BackCompatibility` measures the performance gap when evaluating older, weaker responses with the refreshed strong-response-trained judge instead of the weak-response-trained judge:

$$\texttt{BackCompatibility} = \mathrm{Acc}_{weak}(J_{strong}) - \mathrm{Acc}_{weak}(J_{weak}). \tag{6}$$

This setting is particularly important for established evaluation pipelines: if an old judge is replaced by a new one while the task remains the same, how much does performance differ? We view this as the *data disadvantage* from changing training data from $\mathcal{R}^{train}_{weak}$ to $\mathcal{R}^{train}_{strong}$ when evaluating on $\mathcal{R}^{test}_{weak}$. A positive `BackCompatibility` indicates that the strong-trained judge outperforms the weak-trained judge on weak responses (good backward-compatibility), while a negative value reflects performance degradation (poor backward-compatibility).

`CompatibilityShift` quantifies the weak-to-strong distribution shift when evaluating older, weaker responses with a strong-response-trained, refreshed judge. As noted in the previous section, the reverse shift (strong-to-weak) can strongly affect judge performance. Here, we measure how the out-of-distribution nature of backward-compatibility impacts the newly trained judge:

$$\texttt{CompatibilityShift} = \mathrm{Acc}_{weak}(J_{strong}) - \mathrm{Acc}_{strong}(J_{strong}). \tag{7}$$

This captures the response-distribution shift opposite to `FutureProof`, i.e., from $\mathcal{R}^{\text{test}}_{\text{strong}}$ to $\mathcal{R}^{\text{test}}_{\text{weak}}$ or *strong-to-weak*. It measures how far a strong-trained judge falls below its potential under in-distribution evaluation. A positive value indicates better relative performance on weak responses, while a negative value indicates degradation.

`BackCompatibility` Findings. In Figure 3a, we visualize the backward compatibility of judge models trained on strong responses for the DeepScaleR dataset. When evaluating on weak responses, there is little drop in absolute performance between judges trained on strong responses and those trained on in-distribution weak responses. While methods involving SFT consistently cause small performance drops, our results show that DPO training can enable newly trained judges to *outperform* weak-judge models. The drop due to incompatibility is smaller than the advantage gained when moving from weak to strong responses, as noted in the `RefreshAdvantage` findings. This indicates that judges trained on newer responses are indeed backward-compatible: they closely mimic the performance of weak-trained judges, even in out-of-distribution settings. Likewise on MMLU-Pro, strongly trained judges perform on par with or better than weak-trained judges when evaluating weak responses, as discussed in Appendix F.2. Thus, combined with our findings in Section 5.1, we conclude that re-training with updated responses is universally beneficial: such refreshed judges are not only much better at evaluating new model responses but can also serve as drop-in replacements for their older counterparts with minimal loss in performance.

`CompatibilityShift` Findings. As shown above, judges trained on strong responses roughly match the performance of those trained on weak responses when evaluating older responses. Despite strong absolute performance, such newer judges are evaluating under a *strong-to-weak* distribution shift; Figure 3b plots the drop in performance due to this shift on DeepScaleR dataset. Here, we observe that across all judges and recipes, judges still experience degradation due to the out-of-distribution nature of evaluation, with the lone exception being SFT-trained Llama3.1-8B. Surprisingly, here, the largest model, finetuned from Mistral-24B, experiences the largest absolute drops across all training recipes. These findings highlight that, while stronger trained judges can serve as appropriate drop-in replacements for weaker judges, distribution shift causes them to underperform relative to their potential. We see the same pattern on MMLU-Pro, where strong-trained judges also degrade under response-distribution shift, as discussed in Appendix F.2. However, on DeepScaleR, compared to the degradation from the weak-to-strong response-distribution shift (as measured by `FutureProof` in Section 5.1), these degradations are relatively smaller. This suggests that the weak-to-strong evaluation response-distribution shift is a harder setting than strong-to-weak, again highlighting the importance of retraining judges on new model responses.

5.3 Can continual training improve future-proofing and backward-compatibility of judge models?

Experimental setup. Sections 5.1 and 5.2 show that training a judge *from scratch* on responses from newer generators is advantageous in evaluations. An alternative is to *continually update* a judge originally trained on older responses by incrementally fine-tuning it on newer, stronger responses. We simulate this continual-learning paradigm by further training J_{weak} on responses from stronger generators, denoting the resulting model as $J_{\text{weak}\to\text{strong}}$ (details in Appendix C). All experiments in this section are restricted to training judges on DeepScaler with DPO due to compute constraints.

To assess the effect of continual training, we evaluate $J_{weak\to strong}$ on both future-proofing and backward-compatibility metrics, comparing its performance against that of the original weakly

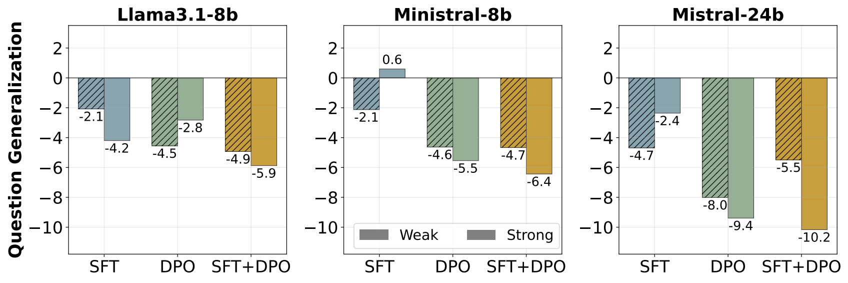


Figure 6: *Question Generalization of DeepScaler-Trained Judges.* Generalization of judges trained on weak vs. strong responses to seen and unseen questions. Judges typically fail to generalize to unseen questions, showing large performance drops relative to evaluating unseen responses on seen questions.

trained judge and the strongly trained judge, respectively. Specifically, we compare `FutureProof` and `RefreshAdvantage` when replacing J_{weak} with $J_{weak \to strong}$ in Equations (4)–(5), as shown in Figure 4. We also compare `CompatibilityShift` and `BackCompat` when replacing J_{strong} with $J_{weak \to strong}$ in Equations (6)–(7), as shown in Figure 5. Together, these comparisons reveal how continual training helps weak judges adapt to future distribution shifts while retaining compatibility with weaker responses, relative to training from scratch.

Changes in Future-Proofing. Figure 4 shows that continual training consistently improves future-proofing. `FutureProof` scores increase across all three models, while `RefreshAdvantage` decreases, approaching zero for Ministral-8B and Mistral-24B. The reduction in `RefreshAdvantage` indicates that the benefit of retraining a strong model from scratch, relative to continual training, largely disappears when evaluating stronger responses. At the same time, the higher `FutureProof` scores of $J_{weak \to strong}$ demonstrate that continual training enables better adaptation to the weak-to-strong distribution shift than simply retaining the weak model.

Changes in Backward-Compatibility. Figure 5 shows mixed but informative results on backward-compatibility. `BackCompatibility` scores increase for Mistral-24B and Llama-3.1-8B but decrease for Ministral-8B. Higher `BackCompatibility` indicates that a continually trained judge remains closer to the weakly trained judge when evaluating weak responses, compared to a model trained solely on strong responses. We also observe a notable increase in `CompatibilityShift`, highlighting that continual training improves adaptation to older, weaker responses relative to purely strong-trained models. Together, these results suggest that continual training can better preserve backward-compatibility in several settings while also enhancing adaptability to distribution shifts.

5.4 How do judges generalize to unseen questions and responses?

Experimental setup. As LLMs advance, both responses and questions evolve (e.g., AIME24 vs. AIME25). We therefore examine how judges perform on previously unseen questions by sampling from $\mathcal{Q}$ in our dual-distribution framework. To quantify the benefits of question exposure during judge training, we define two evaluation splits. In the first, we select a subset of training questions and sample new responses for them, which we call the *seen-questions, unseen-responses* split. In the second, we draw questions from $\mathcal{Q}^{\text{train}}$ that were excluded from training and pair them with new responses, defining the *unseen-questions, unseen-responses* split. Comparing judge performance across these splits reveals the performance gap due to question generalization.

$$\texttt{QuestionGen}_{weak} = \text{Acc}_{weak,unseen}(J_{weak}) - \text{Acc}_{weak,seen}(J_{weak}) \tag{8}$$
$$\texttt{QuestionGen}_{strong} = \text{Acc}_{strong,unseen}(J_{strong}) - \text{Acc}_{strong,seen}(J_{strong}). \tag{9}$$

These metrics capture how well judges generalize across *questions*: responses are drawn from the same generator, with only the question split (seen vs. unseen during training) varied. A positive value of `QuestionGen` indicates better performance on unseen questions, while a negative value indicates failure to generalize to unseen questions.

`QuestionGen` Findings. As shown in Figure 6, current judge models do not generalize well to unseen questions, with nearly all judges exhibiting performance drops compared to evaluating on seen questions with unseen responses. Surprisingly, we find that SFT enables the best generalization, with SFT-trained judges showing the smallest absolute drops in most cases. Mistral-24B, however, exhibits the largest drops within each training recipe, indicating poorer generalization compared to smaller models. These trends are consistent with the corresponding findings on MMLU-Pro, discussed in detail in Appendix F.3. Overall, our experiments reveal that exposing judges to the questions they are likely to evaluate can lead to significant performance gains.

6 Conclusion

We present a dual-distribution framework for automatic evaluation and analyze four key questions surrounding finetuned LLM-as-judge models, a crucial component of the LLM development cycle. First, we study future-proofing and show that judges trained on older responses struggle to evaluate outputs from newer, stronger LLMs, but re-training on newer responses yields substantial gains. Second, we examine backward-compatibility and find that judges trained on newer responses incur only minor drops, or even improvements, when evaluating older responses. Third, we demonstrate that continual learning provides a more balanced adaptation to both older and newer response distributions compared to training solely on stronger or weaker responses. Finally, we investigate question generalization and find that judges experience large drops in performance on questions unseen during training. Overall, our work highlights critical challenges and actionable strategies for developing robust, future-proof, and backward-compatible judge models.

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Francis Christiano, John Schulman, and Dandelion Mané. Concrete problems in ai safety. *ArXiv*, abs/1606.06565, 2016. URL `https://api.semanticscholar.org/CorpusID:10242377`.

Axolotl maintainers and contributors. Axolotl: Post-training for ai models, 2023. URL `https://github.com/axolotl-ai-cloud/axolotl`.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas R. Joglekar, Jan Leike, Ilya Sutskever, Jeff Wu, and OpenAI. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *ArXiv*, abs/2312.09390, 2023. URL `https://api.semanticscholar.org/CorpusID:266312608`.

Nuo Chen, Zhiyuan Hu, Qingyun Zou, Jiaying Wu, Qian Wang, Bryan Hooi, and Bingsheng He. Judgelrm: Large reasoning models as a judge. *ArXiv*, abs/2504.00050, 2025a. URL `https://api.semanticscholar.org/CorpusId:277467872`.

Wei-Lin Chen, Zhepei Wei, Xinyu Zhu, Shi Feng, and Yu Meng. Do llm evaluators prefer themselves for a reason? *arXiv preprint arXiv:2504.03846*, 2025b.

Xiusi Chen, Gaotang Li, Ziqi Wang, Bowen Jin, Cheng Qian, Yu Wang, Hongru Wang, Yu Zhang, Denghui Zhang, Tong Zhang, Hanghang Tong, and Heng Ji. Rm-r1: Reward modeling as reasoning. *ArXiv*, abs/2505.02387, 2025c. URL `https://api.semanticscholar.org/CorpusID:278327900`.

Karl Cobbe, Vineet Kosaraju, Mo Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168, 2021. URL `https://api.semanticscholar.org/CorpusID:239998651`.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony S. Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aur'elien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cris tian Cantón Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab A. AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriele Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guanglong Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Ju-Qing Jia, Kalyan Vasuden Alwala, K. Upasani, Kate Plawiak, Keqian Li, Ken-591 neth Heafield, Kevin R. Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuen ley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melissa Hall Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Niko lay Bashlykov, Nikolay Bogoychev, Niladri S. Chatterji, Olivier Duchenne, Onur cCelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasić, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ron nie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sa hana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Chandra Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Vir ginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whit ney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yiqian Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zhengxu Yan, Zhengxing Chen, Zoe Papakipos, Aaditya K. Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adi Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Po-Yao (Bernie) Huang, Beth Loyd, Beto de Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Shang-Wen Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzm'an, Frank J. Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory G. Sizov, Guangyi Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Han Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kaixing(Kai) Wu, U KamHou, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, A Lavender, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthias Lennie, Matthias

Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Mun ish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navy ata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pe dro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollár, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Kumar Gupta, Sung-Bae Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Andrei Poenaru, Vlad T. Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xia Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models. *ArXiv*, abs/2407.21783, 2024. URL `https://api.semanticscholar.org/CorpusID:271571434`.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. *ArXiv*, abs/2305.14387, 2023. URL `https://arxiv.org/pdf/2305.14387.pdf`.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HJz6tiCqYm`.

Xinyu Hu, Li Lin, Mingqi Gao, Xunjian Yin, and Xiaojun Wan. Themis: A reference-free nlg evaluation language model with flexibility and interpretability. *arXiv preprint arXiv:2406.18365*, 2024.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*, 2023.

Hawon Jeong, chaeHun Park, Jimin Hong, and Jaegul Choo. The comparative trap: Pairwise comparisons amplifies biased preferences of llm evaluators. 2024. URL `https://api.semanticscholar.org/CorpusID:270562681`.

Gemma Team Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ram'e, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gael Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Róbert Istvan Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, Andr'as Gyorgy, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Boxi Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, Cj Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Pluci'nska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, J. Michael Wieting, Jonathan Lai, Jordi Orbay, Joe Fernandez, Joshua Newlan, Junsong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stańczyk, Pouya Dehghani Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Ardeshir Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vladimir Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab S. Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam M. Shazeer, Oriol Vinyals, Jeffrey Dean, Demis Hassabis, Koray Kavukcuoglu, Clément Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and L'eonard Hussenot. Gemma 3 technical report. *ArXiv*, abs/2503.19786, 2025. URL `https://api.semanticscholar.org/CorpusID:277313563`.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*, 2024a. URL `https://openreview.net/forum?id=8euJaTveKw`.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. *ArXiv*, abs/2405.01535, 2024b. URL `https://api.semanticscholar.org/CorpusID:269502688`.

Seungone Kim, Ian Wu, Jinu Lee, Xiang Yue, Seongyun Lee, Mingyeong Moon, Kiril Gashteovski, Carolin Lawrence, J. Hockenmaier, Graham Neubig, and S. Welleck. Scaling evaluation-time compute with reasoning models as process evaluators. *ArXiv*, abs/2503.19877, 2025. URL `https://api.semanticscholar.org/CorpusId:277313538`.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. Wilds: A benchmark of in-the-wild distribution shifts. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5637–5664. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/koh21a.html`.

Michael Krumdick, Charles Lovering, Varshini Reddy, Seth Ebner, and Chris Tanner. No free labels: Limitations of llm-as-a-judge without human grounding. *arXiv preprint arXiv:2503.05061*, 2025.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. Llms-as-judges: A comprehensive survey on llm-based evaluation methods. *ArXiv*, abs/2412.05579, 2024a. URL `https://api.semanticscholar.org/CorpusID:274596907`.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, hai zhao, and Pengfei Liu. Generative judge for evaluating alignment. In *The Twelfth International Conference on Learning Representations*, 2024b. URL `https://openreview.net/forum?id=gtkFw6sZGS`.

Yang Liu, Dan Iter, Yichong Xu, Shuo Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://arxiv.org/pdf/2303.16634.pdf.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TG8KACxEON.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=4NJBV6Wp0h.

Junsoo Park, Seungyeon Jwa, Meiying Ren, Daeyoung Kim, and Sanghyuk Choi. Offsetbias: Leveraging debiased data for tuning evaluators. *ArXiv*, abs/2407.06551, 2024a. URL https://arxiv.org/pdf/2407.06551.pdf.

Junsoo Park, Seungyeon Jwa, Meiying Ren, Daeyoung Kim, and Sanghyuk Choi. Offsetbias: Leveraging debiased data for tuning evaluators. *arXiv preprint arXiv:2407.06551*, 2024b.

Pouya Pezeshkpour and Estevam Hruschka. Large language models sensitivity to the order of options in multiple-choice questions. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2006–2017, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.130. URL https://aclanthology.org/2024.findings-naacl.130/.

Vyas Raina, Adian Liusie, and Mark J. F. Gales. Is llm-as-a-judge robust? investigating universal adversarial attacks on zero-shot llm assessment. *ArXiv*, abs/2402.14016, 2024. URL https://api.semanticscholar.org/CorpusId:267770121.

Gemma Team Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, L'eonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ram'e, Johan Ferret, Peter Liu, Pouya Dehghani Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stańczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Boxi Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Christoper A. Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozi'nska, D. Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Pluci'nska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost R. van Amersfoort, Josh Gordon, Josh Lipschultz, Joshua Newlan, Junsong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, L. Sifre, Lena Heuermann, Leti cia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Gorner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nen shad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Peng chong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Perrin, Sébastien M. R. Arnold, Se bastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomás Kociský, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeffrey Dean, Demis Hassabis, Koray Kavukcuoglu, Clément Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size. *ArXiv*, abs/2408.00118, 2024. URL https://api.semanticscholar.org/CorpusID:270843326.

Jon Saad-Falcon, Rajan Vivek, William Berrios, Nandita Shankar Naik, Matija Franklin, Bertie Vidgen, Amanpreet Singh, Douwe Kiela, and Shikib Mehri. Lmunit: Fine-grained evaluation with natural language unit tests. *arXiv preprint arXiv:2412.13091*, 2024.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Comput. Surv.*, May 2025. ISSN 0360-0300. doi: 10.1145/3735633. URL https://doi.org/10.1145/3735633. Just Accepted.

Nishad Singhi, Hritik Bansal, Arian Hosseini, Aditya Grover, Kai-Wei Chang, Marcus Rohrbach, and Anna Rohrbach. When to solve, when to verify: Compute-optimal problem solving and generative verification for llm reasoning. *ArXiv*, abs/2504.01005, 2025. URL https://api.semanticscholar.org/CorpusId:277467695.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3008–3021. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1f89885d556929e98d3ef9b86448f951-Paper.pdf.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. Easy-to-hard generalization: Scalable alignment beyond human supervision. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=qwgfh2fTtN.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9275–9293, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.746. URL https://aclanthology.org/2020.emnlp-main.746/.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Yuan Tang, Alejandro Cuadron, Chenguang Wang, Raluca Popa, and Ion Stoica. Judgebench: A benchmark for evaluating LLM-based judges. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=G0dksFayVq.

Mistral Team. Un ministral, des ministraux, a. URL https://mistral.ai/news/ministraux.

Mistral Team. Mistral small 3, b. URL https://mistral.ai/news/mistral-small-3.

Tuhina Tripathi, Manya Wadhwa, Greg Durrett, and Scott Niekum. Pairwise or pointwise? evaluating feedback protocols for bias in LLM-based evaluation. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=uyX5Vnow3U.

Tu Vu, Kalpesh Krishna, Salaheddin Alzubi, Chris Tar, Manaal Faruqui, and Yun-Hsuan Sung. Foundational autoraters: Taming large language models for better automatic evaluation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17086–17105, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.949. URL https://aclanthology.org/2024.emnlp-main.949/.

Peifeng Wang, Austin Xu, Yilun Zhou, Caiming Xiong, and Shafiq Joty. Direct judgement preference optimization. *arXiv preprint arXiv:2409.14664*, 2024a.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. *ArXiv*, abs/2305.17926, 2023. URL https://api.semanticscholar.org/CorpusID:258960339.

Tianlu Wang, Ilia Kulikov, Olga Golovneva, Ping Yu, Weizhe Yuan, Jane Dwivedi-Yu, Richard Yuanzhe Pang, Maryam Fazel-Zarandi, Jason Weston, and Xian Li. Self-taught evaluators. *arXiv preprint arXiv:2408.02666*, 2024b.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290. Curran Associates, Inc., 2024c. doi: 10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ad236edc564f3e3156e1b2feafb99a24-Paper-Datasets_and_Benchmarks_Track.pdf.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. J1: Incentivizing thinking in llm-as-a-judge via reinforcement learning. *arXiv preprint arXiv:2505.10320*, 2025.

Austin Xu, Srijan Bansal, Yifei Ming, Semih Yavuz, and Shafiq Joty. Does context matter? contextualjudgebench for evaluating llm-based judges in contextual settings. *arXiv preprint arXiv:2503.15620*, 2025a.

Austin Xu, Yilun Zhou, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. J4r: Learning to judge with equivalent initial state group relative policy optimization. *ArXiv*, abs/2505.13346, 2025b. URL https://api.semanticscholar.org/CorpusID:278768650.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. Qwen2 technical report. *ArXiv*, abs/2407.10671, 2024a. URL https://api.semanticscholar.org/CorpusID:271212307.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024b. URL https://api.semanticscholar.org/CorpusID:274859421.

Ziyi Ye, Xiangsheng Li, Qiuchi Li, Qingyao Ai, Yujia Zhou, Wei Shen, Dong Yan, and Yiqun Liu. Beyond scalar reward model: Learning generative judge from preference data. *arXiv preprint arXiv:2410.03742*, 2024.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason E. Weston. Self-rewarding language models. *ArXiv*, abs/2401.10020, 2024. URL `https://arxiv.org/pdf/2401.10020.pdf`.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. Evaluating large language models at evaluating instruction following. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=tr0KidwPLc`.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL `https://openreview.net/forum?id=uccHPGDlao`.

Yilun Zhou, Austin Xu, PeiFeng Wang, Caiming Xiong, and Shafiq Joty. Evaluating judges as evaluators: The jetts benchmark of llm-as-judges as test-time scaling evaluators. *ArXiv*, abs/2504.15253, 2025. URL `https://api.semanticscholar.org/CorpusId:277955867`.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. JudgeLM: Fine-tuned large language models are scalable judges. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=xsELpEPn4A`.

A LLM Usage

Other than being used as part of the experiments conducted in this work, LLMs were used solely as a writing assistance tool in preparing this paper submission. Their role was limited to polishing language, improving clarity, and reducing redundancy. The prompt used for this purpose was similar to "Please revise the writing of this, making sure to remove any grammatical mistakes." All research ideas, experimental designs, analyses, and claims presented in the paper are entirely the original work of the authors. No part of the conceptual, methodological, or empirical contributions relies on or originates from LLM outputs.

B Generators and Generator Strengths

Shorthand	**Full Hugging Face Identifier**
Llama3.3-70B	`meta-llama/Llama-3.3-70B-Instruct`
Llama3.1-8B	`meta-llama/Llama-3.1-8B-Instruct`
Qwen2-7B	`Qwen/Qwen2-7B-Instruct`
Qwen2.5-7B	`Qwen/Qwen2.5-7B-Instruct`
Qwen2.5-14B	`Qwen/Qwen2.5-14B-Instruct`
Qwen2.5-32B	`Qwen/Qwen2.5-32B-Instruct`
Gemma2-9B	`google/gemma-2-9b-it`
Gemma3-12B	`google/gemma-3-12b-it`
Ministral-8B	`mistralai/Ministral-8B-Instruct-2410`
Mistral-Small-24B	`mistralai/Mistral-Small-24B-Instruct-2501`
DeepScaleR	`agentica-org/DeepScaleR-Preview-Dataset`
MMLU-Pro	`TIGER-Lab/MMLU-Pro`

Table 1: Mapping from shorthand model and dataset names to their corresponding Hugging Face identifiers.

To curate generator responses, we begin with a set of candidate generators and a collection of questions Q, along with verifiable ground-truth answers $A^{\star}$. For each question, we sample 20 responses from each generator using temperature sampling and compute a *pass@1* score. This score represents

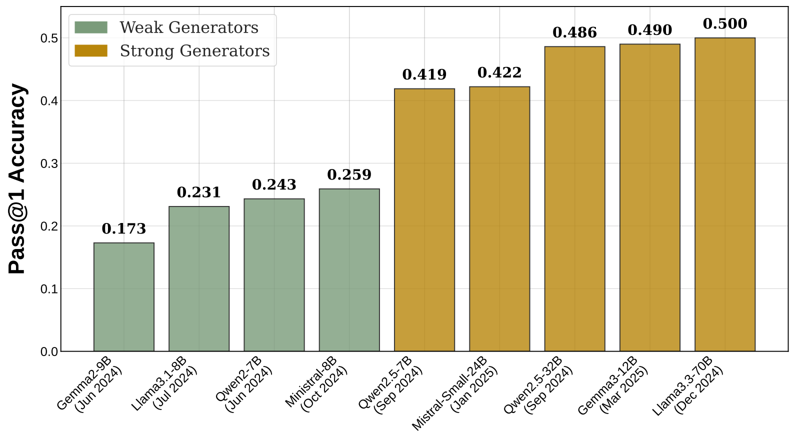


Figure 7: Generator strength on the DeepScaleR dataset, measured using pass@1 with 20 independently sampled responses. Models fall into two well-separated strength clusters: weak (0.17–0.26) and strong (0.42–0.50). No models occupy the 0.26–0.42 gap (a 0.16-wide gap), making the clustering robust to any threshold chosen within this interval. This clustering also aligns with model release dates, with stronger, newer models (yellow) outperforming weaker, older ones (green).

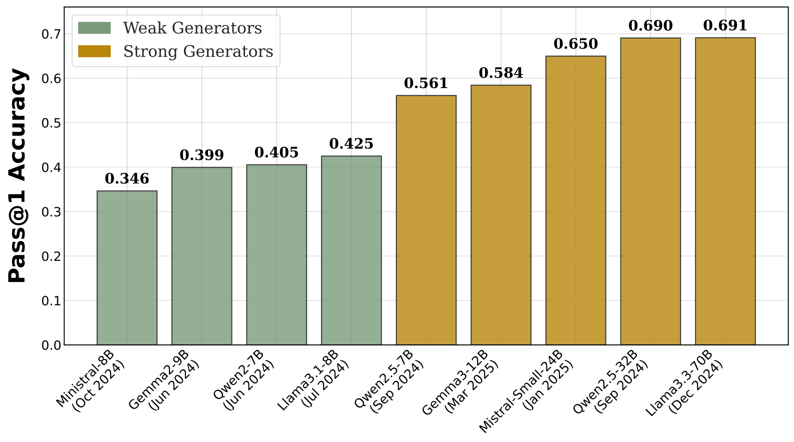


Figure 8: Generator strength on the MMLU-Pro dataset, measured using pass@1 with 20 independently sampled responses. The same two-tier structure appears: weak (0.34–0.43) vs. strong (0.56–0.69), with no models in the 0.43–0.56 intermediate gap (a 0.13-wide gap). This clustering aligns with the weak–strong clustering observed in DeepScaleR (Figure 7), indicating that the clustering reflects underlying model strength rather than threshold choice or dataset-specific artifacts.

the probability of obtaining at least one correct solution when randomly selecting one solution from the 20 attempts, where correctness is determined by matching the generator's responses against $A^\star$.

Concretely, we use two verifiable datasets, as shown in Table 1: DeepScaleR (Luo et al., 2025) and MMLU-Pro (Wang et al., 2024c). DeepScaleR contains 40K challenging Olympiad-level math problems spanning multiple years, each paired with a ground-truth answer. In contrast, MMLU-Pro contains 12K multiple-choice problems, spanning 57 subjects across 14 categories and drawing from diverse sources such as MMLU, STEM websites, TheoremQA, and SciBench. We include MMLU-Pro to demonstrate the broader applicability of our results beyond mathematics. For all experiments, we use popular open-source instruction-tuned models, as listed in Table 1. Gemma-2-9B (Riviere et al., 2024), Gemma-3-12B (Kamath et al., 2025), Llama-3.1-8B, Llama-3.3-70B Dubey et al. (2024), Ministral-8B (Team, a), and Mistral-Small-24B (Team, b), Qwen2-7B (Yang et al., 2024a), Qwen2.5-7B (Yang et al., 2024b), and Qwen2.5-32B (Yang et al., 2024b).

Judge Backbone LLM	Weak Response Dataset	Strong Response Dataset
Ministral-8B, Mistral-Small-24B	Gemma2-9B, Qwen2-7B, Llama3.1-8B	Qwen2.5-7B, Gemma3-12B, Llama3.3-70B
Llama3.1-8B	Gemma2-9B, Qwen2-7B, Ministral-8B	Qwen2.5-7B, Gemma3-12B, Mistral-Small-24B

Table 2: Overview of training data composition on a per-backbone LLM basis. To mitigate bias from the difficulty of evaluating self-generated responses, we avoid training judge models on their own responses. This produces per-judge training datasets composed of different generator responses.

Figure 7, which plots the pass@1 scores of all candidate generators, reveals two clearly separated capability bands on DeepScaleR. Weak models fall in the 0.17–0.26 range, whereas strong models fall in the 0.42–0.50 range, leaving a 0.16-wide empty gap (0.26–0.42) with no model in the intermediate region. Thus, any threshold chosen within this interval produces the same weak-strong grouping. This separation also aligns well with model release dates, as shown in Figure 7. We observe the same two-tier pattern on MMLU-Pro, as shown in Figure 8: weak models score 0.34–0.43, while strong models score 0.56–0.69, again with no models occupying the 0.43–0.56 interval (a 0.13-wide gap). The alignment of weak–strong groups across two very different datasets indicates that the distinction captures genuine differences in underlying model strength, rather than artifacts of a particular dataset or threshold choice.

C Training Setup Details

Dataset Construction. To create the training and evaluation splits, we first construct pairwise input samples for the judge, following prior work (Tan et al., 2025; Wang et al., 2024b). For each question, we sample multiple responses from each generator, and each response is then labeled as "correct" or "incorrect" according to the ground-truth answer $A^\star$. We then form response pairs, where each pair consists of one correct response and one incorrect response, resulting in a pairwise sample with an objectively correct answer. Importantly, responses in a pair are drawn from a single generator only. This choice ensures that the judge learns to distinguish correctness based on reasoning quality rather than relying on stylistic differences between models, which could occur if responses from different generators were mixed in a single pair. For each generator and question, we only keep samples where there is at least one correct and one incorrect response and if this condition is not met, the question is discarded for that generator. In Table 3, we report the percentage of questions retained for each generator after applying this discarding criterion. Further, in Figure 9, we show that weak models discard many hard questions because all 20 samples are incorrect, whereas strong models discard many easy questions because all 20 samples are correct. Mid-tier models retain the most questions because they more frequently produce mixed outcomes, resulting in a clear U-shaped trend in the rank–retention plot. Thus, the retained subset is enriched for borderline questions near each model's decision boundary, naturally inducing a medium-difficulty selection bias. Following this, and based on the generator strengths defined in Appendix B, we construct aggregated pairwise datasets consisting exclusively of either weak or strong responses, which we refer to as the *weak dataset* and *strong dataset*, respectively.

Judge Data Distillation & Training Objectives. We train judges using three commonly adopted recipes: supervised fine-tuning (SFT) (Li et al., 2024b; Kim et al., 2024a; Vu et al., 2024), direct preference optimization (DPO) (Hu et al., 2024; Wang et al., 2024b), and a combined SFT and DPO objective (Wang et al., 2024a; Ye et al., 2024; Saad-Falcon et al., 2024). As these recipes require supervision, specifically, the CoT explanation C (Sec. 2), we adopt the common *teacher model* convention (Li et al., 2024b; Wang et al., 2024a). We prompt GPT-4o with (Q, R_1, R_2) inputs, sampling multiple responses $(C, \hat{V})$ per input. Based on the ground-truth verdict V^*, we categorize responses as correct (positive) samples y^+ or incorrect (negative) samples y^-. We only keep inputs for which at least one y^+ and y^- exists. This ensures that the inputs are exactly comparable for SFT and DPO. Positive samples are then used for SFT, whereas positive-negative pairs are used for DPO-based recipes.

Generator	**DeepScaleR** $\text{Ret\%}_{\text{rank}}$	**MMLU-Pro** $\text{Ret\%}_{\text{rank}}$
Gemma-2-9B	36.1_1	63.83_2
Gemma-3-12B	57.2_8	49.23_6
Qwen-2-7B	52.2_3	79.31_3
Qwen-2.5-7B	63.7_5	58.75_5
Qwen-2.5-32B	62.9_7	43.61_8
Llama-3.1-8B	52.2_2	76.38_4
Llama-3.3-70B	47.1_9	34.29_9
Ministral-8B	62.5_4	62.32_1
Mistral-Small-24B	64.6_6	50.73_7

Table 3: Retention percentage (Ret%) across DeepScaleR and MMLU-Pro for various generators. The subscript rank denotes each model's Pass@1 rank; higher ranks correspond to models with superior performance, as shown in Figure 7 and Figure 8. Retention measures the fraction of questions where a generator produces both a correct and an incorrect sample across 20 generations.

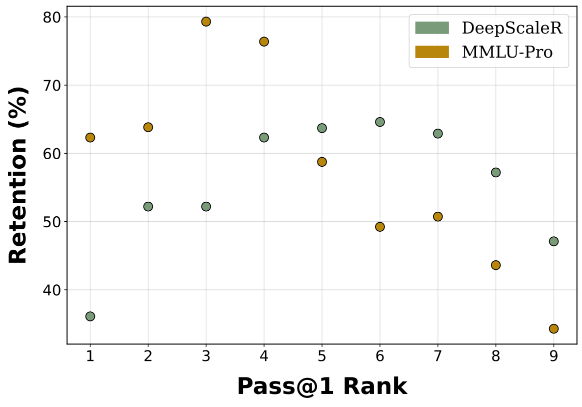


Figure 9: Retention vs. Pass@1 rank, derived from Table 3. Weak generators drop hard questions because all sampled responses are incorrect, strong generators drop easy ones because all sampled responses are correct, and mid-tier generators retain the most by producing responses with mixed correctness. This concentrates the retained questions on borderline, medium-difficulty items near each generator's decision boundary.

Train and Evaluation Splits. To analyze the four practical questions described in Section 1 using the dual-distribution framework from Section 3, we split the weak and strong datasets into training and test splits. For testing, we construct two distinct splits: an *unseen-questions* split and a *seen-questions* split. The unseen-questions split contains questions not present during training, while the seen-questions split reuses training questions but samples *new* responses, with pairs constructed following the same process as described above. Unless otherwise specified, we use the unseen-questions split for evaluation. Note that the corresponding weak and strong splits use exactly the same set of questions; we remove any question that appears in only one split. This prevents question-difficulty differences from confounding our findings. Overall, for DeepScaleR each training split contains 70K samples, whereas for MMLU-Pro each contains 10K samples. For both datasets, each evaluation split includes 2.5K response-order-unflipped samples (5K after response-order flips).

Generator and Judge Backbone Details. We choose three backbone models to finetune: Llama-3.1-8B, Ministral-8B, and Mistral-24B, covering a range of model sizes and intrinsic strengths. Prior work (Tan et al., 2025) has shown that models often struggle to judge the correctness of pairs of their own sampled responses. Another line of work (Chen et al., 2025b; Panickssery et al., 2024) has shown that models can recognize their own responses and exhibit self-bias. Thus, to disentangle any effects of training a judge on self-generated responses, we exclude the backbone judge model from serving as a generator. Specifically, we create two training sets (each with weak and strong splits), ensuring that the backbone judge model is not included in the list of generators. We summarize these training sets and the associated backbone models in Table 2.

Hyperparameters. All experiments with SFT, DPO, SFT+DPO are implemented using the AXOLOTL framework Axolotl maintainers and contributors (2023). For SFT, we sweep learning rates in $\{1 \times 10^{-6}, 2.5 \times 10^{-6}, 5 \times 10^{-6}, 1 \times 10^{-5}\}$ with a cosine decay scheduler. Across all evaluation splits, a learning rate of 2.5×10^{-6} consistently yields the best performance. For DPO, we adopt standard hyperparameter choices from prior work (Ivison et al., 2023), using a learning rate of 5×10^{-7} and a preference strength parameter $\beta = 0.1$. For SFT+DPO, we optimize a joint loss with equal weighting between the SFT and DPO objectives, using the same DPO hyperparameters (learning rate 5×10^{-7}, $\beta = 0.1$). DeepScaler weak and strong judges are trained for 3 epochs (2,800 gradient steps). In contrast, MMLU-Pro weak and strong judges are trained for 10 epochs (1,500 gradient steps). For continual training experiments (section 5.3), we start from a weak-response DPO-trained judge (trained for 3 epochs) and further train it on strong responses for 1 additional epoch, amounting to roughly 1,000 additional gradient steps. We sweep $\beta \in \{0.1, 1.0\}$ and report results in the main text using $\beta = 1.0$; additional results are included in Table 4 and in Appendix D.

D Consistent Accuracy and Judge's Performance Across splits

Consistent Accuracy. Since judge models are prone to positional biases (Wang et al., 2023; Li et al., 2024b; Xu et al., 2025b)—where their preference shifts depending on whether R_1 or R_2 appears first in the prompt—it is standard practice to evaluate judges using both response orderings (Tan et al., 2025; Xu et al., 2025a;b). Concretely, for input $x = (Q, R_1, R_2)$, let $\bar{x}$ denote the same sample, but with response order flipped in the input prompt, i.e., $\bar{x} = (Q, R_2, R_1)$. Then, evaluation with *consistent accuracy* considers the judge correct only if it correctly identifies the better response under both orderings:

$$\text{Acc} = \frac{1}{|P|} \sum_{x \in P} \mathbb{1}[\hat{V}(x) = V^*(x) \wedge \hat{V}(\bar{x}) = V^*(\bar{x})], \tag{10}$$

where $\mathbb{1}[\cdot]$ is the indicator function, P is the evaluation set consisting of pairs $(x, V^\star(x))$, and the judge's verdicts $\hat{V}(x)$ are compared against the ground-truth verdicts $V^\star(x)$.

Judge's Performance. We report all consistent-accuracy scores of our trained judges for both DeepScaler and MMLU-Pro across the different evaluation splits in Table 4.

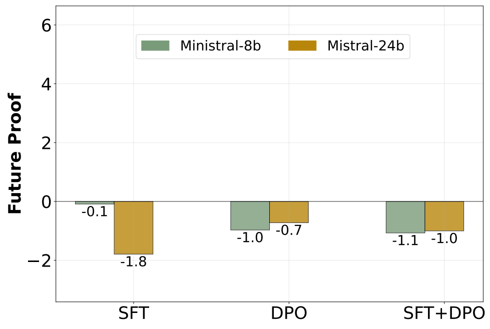
E Research Questions in the Dual-Distribution Formulation

As described in Section 3, the dual-distribution formulation separates the *question distribution* $\mathcal{Q}$ from the *response distribution* $\mathcal{R}$, reflecting two real-world sources of shift: (1) more capable generators (an evolving $\mathcal{R}$) and (2) new questions (an evolving $\mathcal{Q}$). This decomposition allows us to isolate and quantify the impact of each factor on judge performance. Building on this, we investigate several practical questions about the *shelf life* of trained judges, focusing on four distinct settings:

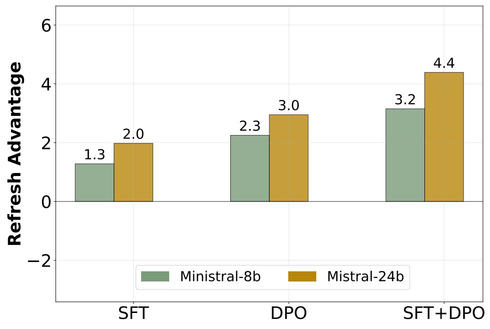
How future-proof are judge models? For a judge to be future-proof, it must be able to evaluate responses from newer, stronger models. To study this, we examine how a judge trained on responses from the current generation of weak models performs when evaluating responses from strong models. Specifically, we train a judge on $\mathcal{R}_{weak}^{train}$ and evaluate it on both $\mathcal{R}_{weak}^{test}$ and $\mathcal{R}_{strong}^{test}$. This setup characterizes how robust judges are to a *distribution shift from weak to strong* responses. Additionally, we quantify the gains from retraining a judge on strong responses by replacing training data from $\mathcal{R}_{weak}^{train}$ with responses from $\mathcal{R}_{strong}^{train}$.

How backward-compatible are judge models? Newly trained judges are fine-tuned to evaluate newer, stronger response-generating models. However, does this focus on state-of-the-art generators come at the expense of performance on older, more established generators? To complement our future-proofing experiments, we examine backward-compatibility. Specifically, given a judge trained on responses from $\mathcal{R}_{strong}^{train}$, we ask: how well does it match a judge trained on weaker responses from $\mathcal{R}_{weak}^{train}$ when both are evaluating $\mathcal{R}_{weak}^{test}$ responses? Beyond this comparison, evaluating weaker responses with a judge trained on strong responses also introduces a *distribution shift from strong to weak* responses. We quantify any performance losses that result from this shift.

Can continual learning improve future-proofing and backward-compatibility of judge models? Rather than training a new judge from scratch on $\mathcal{R}_{strong}$, we start with a judge trained on $\mathcal{R}_{weak}^{train}$

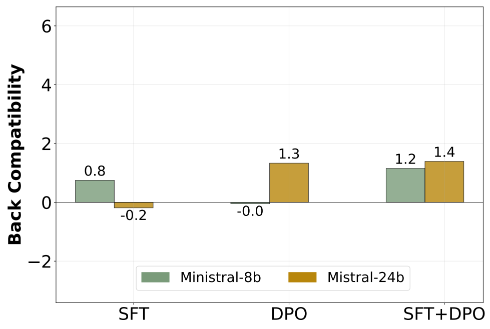


(a) `FutureProof.`

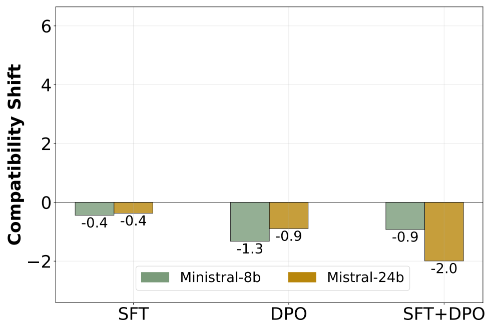


(b) `RefreshAdvantage.`

Figure 10: *Future-proofing of MMLU-Pro–trained judges.* (a) Future-proofing measured by `FutureProof`; negative values indicate degraded performance on stronger responses. All models and training recipes degrade, reflecting poor evaluation of newer, stronger responses. (b) Benefits of re-training on strong responses, measured by `RefreshAdvantage`. Re-training consistently improves performance, with larger gains under DPO-based recipes. These results largely follow the trends observed on DeepScaler (Figure 10), except with smaller absolute magnitudes, indicating that response-distribution shift can depend on the domain.



(a) `BackCompatibility.`



(b) `CompatibilityShift.`

Figure 11: *Backward-Compatibility of MMLU-Pro-Trained Judges.* (a) `BackCompatibility` measures how well judges trained on strong responses evaluate older responses; positive values indicate improvements over weak-judge baselines. Strong-trained judges show clear gains, larger than those on DeepScaler Figure 3a, suggesting that strong judges are as good as or better than weak judges when evaluating weak responses. (b) Despite strong absolute performance, newer judges still face distribution shift, reflected in `CompatibilityShift`, which captures performance drops relative to evaluating strong responses. These shifts are similar to those observed on DeepScaler Figure 3b.

and continually fine-tune it on $\mathcal{R}^{train}_{strong}$ to obtain a continually trained judge. In parallel to the settings above, we ask whether the continually trained judge narrows the gap on $\mathcal{R}^{test}_{strong}$ relative to one trained only on $\mathcal{R}^{train}_{weak}$, and whether it retains performance on $\mathcal{R}^{test}_{weak}$ relative to a judge trained from scratch on $\mathcal{R}^{train}_{strong}$. This setup tests whether continual training helps a weak judge adapt to the weak to strong response shift while preserving compatibility with older responses.

How do judges generalize across unseen questions? As new questions are introduced for evaluating LLMs, judge models must accurately assess responses to these questions. Here, we quantify the benefit of a judge model having seen a question during training. To study this form of *generalization*, we construct two evaluation splits. The first is a *seen-questions, unseen-responses* split, created by selecting questions that appeared in the training set and sampling a new set of responses for these questions from $\mathcal{R}^{train}$. The second is an *unseen-questions, unseen-responses* split, generated by sampling questions from $\mathcal{Q}^{train}$ that were not included in the training data, along with their corresponding responses from $\mathcal{R}^{train}$. Comparing performance across these splits enables us to assess how well judges generalize to previously seen questions versus entirely new ones.

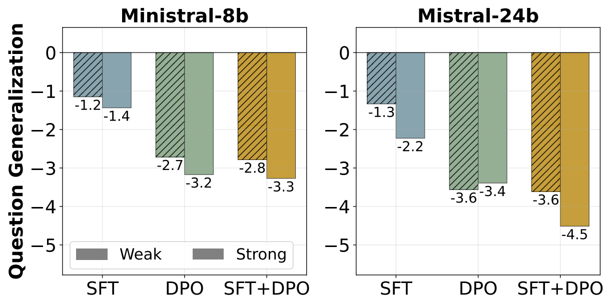


Figure 12: *Question Generalization of MMLU-Pro-Trained Judges.* Generalization of judges trained on weak and strong responses to seen and unseen questions. Judges consistently fail to generalize to unseen questions, showing large performance drops relative to their performance on seen questions. These trends align with our findings on DeepScaler dataset in Figure 6.

F Detailed Findings from MMLU-Pro Dataset

In this section, we present the future-proofing, backward-compatibility, and question-generalization results for the MMLU-Pro dataset and place them in context with the corresponding findings on DeepScaler. As discussed in Section 5, the overall trends on MMLU-Pro closely match those observed on DeepScaler. However, compared to DeepScaler, which is math-oriented and reasoning-intensive, MMLU-Pro exhibits noticeably smaller degradations across all metrics. Since MMLU-Pro is more knowledge-centered, this suggests that the severity of response-distribution shift is domain dependent. These observations imply that a judge model's shelf-life metrics can vary meaningfully with task domain, even when the training recipe and backbone model are held constant. Below, we describe the results for each metric on MMLU-Pro.

F.1 How future-proof are judge models?

`FutureProof` Findings. Figure 10a reports the `FutureProof` values for all models and training recipes. Consistent with DeepScaler, we do not observe any case where judges generalize to newer or stronger responses: all `FutureProof` values are negative. However, the magnitudes on MMLU-Pro are noticeably smaller than those on DeepScaler (see Figure 2a), suggesting that degradation under response-distribution shift is less severe on knowledge-oriented, non-math tasks than on reasoning-intensive math-olympiad problems. This highlights that the extent of future-proofing failure can vary by domain.

`RefreshAdvantage` Findings. As shown in Figure 10b, re-training on up-to-date responses consistently improves evaluation performance: all training recipes and backbone models exhibit positive `RefreshAdvantage` values. Mirroring DeepScaler, DPO-based recipes yield larger gains than SFT alone, and the benefits grow with judge model size. For instance, under SFT+DPO, Mistral-24B gains 4.4 absolute points, compared to 3.2 points for its 8B counterpart, Mistral-8B. Overall, these results reinforce the DeepScaler observation that reliably evaluating stronger generators requires judges trained on strong, contemporary responses. However, the gains on MMLU-Pro are slightly weaker compared to DeepScaler, again indicating that the absolute advantage from refreshing is domain-dependent.

F.2 How backward-compatible are judge models?

`BackCompatibility` Findings. In Figure 11a, we visualize the backward-compatibility of judge models trained on strong responses. When evaluated on weak responses, these judges show little improvement over judges trained directly on in-distribution weak responses. In comparison, on DeepScaler (see Section 5.2), we observed a minimal performance drop, with the `BackCompatibility` metric slightly negative. Taken together, these results indicate that judges trained on newer, stronger responses are indeed backward-compatible: they perform on par with weak-trained judges, even when evaluated out of distribution.

`CompatibilityShift` Findings. Our findings in `BackCompatibility` show that judges trained on strong responses perform comparably to, or better than, weak-trained judges when scoring older responses. However, these stronger judges are evaluated under a *strong-to-weak* distribution mismatch, and Figure 11b illustrates the resulting drop in accuracy. These drops are consistent across models and training recipes. Thus, even though stronger judges can effectively replace weaker ones, distribution shift still limits their realized performance. We observed a similar pattern on the DeepScaleR dataset, as discussed in Section 5.2.

F.3 How do judges generalize to unseen questions and responses?

`QuestionGen` Findings. From Figure 12, we observe that judges trained on MMLU-Pro do not generalize well to unseen questions, with nearly all judges showing performance drops compared to evaluating on seen questions with unseen responses. The trends are similar to those for DeepScaleR in Figure 6: more performant judges using the DPO recipe and larger backbones such as Mistral-24B exhibit larger drops when evaluated on in-distribution questions not encountered during training.

G Prompts and Sampling Hyperparameters

To obtain generator responses, we sample multiple completions from each generator in order to better capture the diversity of its reasoning behaviors. We use five temperature–sampling configurations, where n denotes the number of sampled completions and t the sampling temperature: $(n{=}1, t{=}0.0)$, $(n{=}4, t{=}0.4)$, $(n{=}5, t{=}0.5)$, $(n{=}5, t{=}0.6)$, and $(n{=}5, t{=}0.7)$, with top-p fixed at 1.0. This yields 20 total responses per question for each generator model.

To reduce prompt-format bias and further increase response diversity, we randomly select one of four generator prompt templates (Prompts 1–4) for each sampled completion in the DeepScaleR dataset. For multi-domain experiments using MMLU-Pro, we use the prompt in Prompt 5.

For the judge models, we provide the original question along with two generator responses, each containing both the intermediate reasoning and the final numerical answer. Judge models are decoded greedily using $(n{=}1, t{=}0.0)$, as we found pass@k judge accuracies to be highly correlated with greedy decoding while being more computationally efficient.

We use the prompt in Prompt 6 for judge evaluation.

Generator Prompt Template 1 — DeepScaleR

```
Instruction:
Solve the following math problem step by step.  The last line of your
response should be:
Answer: $Answer
where $Answer is the final answer.
Problem:
{{problem}}
Output Format:
Answer: <your answer here>
```

Prompt 1: Generator prompt template used in DeepScaleR for structured, step-by-step solutions.

Generator Prompt Template 2 — DeepScaleR

```
Instruction:
Solve the following math problem efficiently and clearly.
- For simple problems (2 steps or fewer): give a concise solution with
minimal explanation.
- For complex problems: use the structured step-by-step format:
   Step 1: [Concise description]
  [Explanation / calculations]
   Step 2: [Concise description]
  [Explanation / calculations]
  ...
Important:
Always conclude with:
Therefore, the final answer is: $\boxed {answer}$.
where answer is the final numeric answer.
Problem:
Problem: {{problem}}
```

Prompt 2: Generator prompt template used in DeepScaleR that adapts to problem complexity, producing either concise explanations or multi-step structured reasoning.

Generator Prompt Template 3 — DeepScaleR

```
Instruction:
Read the problem, reason through it, and provide a final answer.
Problem:
{{problem}}
Output Requirement:
Your response must end with:
The final answer is [answer]
where [answer] is the final computed answer.
```

Prompt 3: Generator prompt template that prompts models to reason and explicitly report a final answer.

Generator Prompt Template 4 — DeepScaleR

```
Problem:
{{problem}}
```

Prompt 4: A minimal generator prompt template presenting only the raw problem.

Generator Prompt Template — MMLU-Pro

```
Instruction:
You are given a multiple-choice question from the domain of {{domain}}.
Each answer option corresponds to a lettered choice.
Question:
{{question}}
Options:
{{options}}
Task:
Provide a careful, step-by-step analysis of the question. Use your
reasoning to evaluate all relevant information and identify the correct
option. After completing your reasoning, produce your final choice in
the following format:
<answer>X</answer>
where X is the letter of the correct option.
```

Prompt 5: Prompt template used for reasoning over multiple-choice questions in MMLU-Pro.

Judge Prompt Template

```
Task:
You are a rigorous evaluator comparing two responses to the same math
question.  Judge which response is better, based solely on logical
soundness and correctness.
You are given:
- A Question
- Response A
- Response B
Evaluation Guidelines:
1.  Correctness is top priority.  Prefer responses with correct
reasoning and correct final answers.
2.  If both have reasoning flaws, choose the one that still reaches the
correct final answer.
3.  Ignore style, length, formatting, verbosity, or fluency.
---
Output Format (JSON):
Your final output must be exactly one of the following:
{"verdict": "A"}
{"verdict": "B"}
---
Question:
{{question}}
Response A:
{{response_a}}
Response B:
{{response_b}}
```

Prompt 6: Judge prompt template used to compare two generator responses based on final-answer correctness and reasoning quality.

		DeepScaler				MMLU-Pro			
Train	**Eval**	**0-Shot**	**SFT**	**DPO**	**SFT+DPO**	**0-Shot**	**SFT**	**DPO**	**SFT+DPO**
					Llama3.1-8B				
J_{Wk}	Wk, Sn	32.44	48.14	43.95	63.40	–	–	–	–
	St, Sn	28.41	44.66	36.62	60.48	–	–	–	–
	Wk, Un	30.79	46.06	39.41	58.47	–	–	–	–
	St, Un	27.76	41.91	33.94	55.29	–	–	–	–
J_{St}	Wk, Sn	32.44	45.33	42.53	61.72	–	–	–	–
	St, Sn	28.41	46.41	43.74	65.15	–	–	–	–
	Wk, Un	30.79	44.12	39.61	57.60	–	–	–	–
	St, Un	27.76	42.21	40.91	59.27	–	–	–	–
$J^{0.1}_{\mathrm{Wk}\to\mathrm{St}}$	Wk, Sn	32.44	–	44.69	–	–	–	–	–
	St, Sn	28.41	–	41.19	–	–	–	–	–
	Wk, Un	30.79	–	40.09	–	–	–	–	–
	St, Un	27.76	–	38.41	–	–	–	–	–
$J^{1.0}_{\mathrm{Wk}\to\mathrm{St}}$	Wk, Sn	32.44	–	45.43	–	–	–	–	–
	St, Sn	28.41	–	39.13	–	–	–	–	–
	Wk, Un	30.79	–	40.07	–	–	–	–	–
	St, Un	27.76	–	37.22	–	–	–	–	–
					Ministral-8B				
J_{Wk}	Wk, Sn	33.87	48.06	61.04	61.39	26.87	34.86	47.24	47.81
	St, Sn	28.72	41.94	55.55	56.41	27.14	34.04	46.38	47.08
	Wk, Un	33.81	45.93	56.41	56.72	27.05	33.72	44.54	45.03
	St, Un	29.14	41.91	54.86	53.26	25.74	33.62	43.56	43.96
J_{St}	Wk, Sn	33.87	45.05	60.60	62.25	26.87	35.20	46.32	48.04
	St, Sn	28.72	43.31	64.69	67.30	27.14	36.34	48.98	50.38
	Wk, Un	33.81	42.62	57.15	58.82	27.05	34.46	44.48	46.18
	St, Un	29.14	43.90	59.15	60.86	25.74	34.90	45.82	47.12
$J^{0.1}_{\mathrm{Wk}\to\mathrm{St}}$	Wk, Sn	33.87	–	62.11	–	–	–	–	–
	St, Sn	28.72	–	60.43	–	–	–	–	–
	Wk, Un	33.81	–	54.67	–	–	–	–	–
	St, Un	29.14	–	53.13	–	–	–	–	–
$J^{1.0}_{\mathrm{Wk}\to\mathrm{St}}$	Wk, Sn	33.87	–	59.24	–	–	–	–	–
	St, Sn	28.72	–	58.51	–	–	–	–	–
	Wk, Un	33.81	–	55.28	–	–	–	–	–
	St, Un	29.14	–	54.84	–	–	–	–	–
					Mistral-24B				
J_{Wk}	Wk, Sn	41.00	52.18	76.57	76.90	38.51	45.14	57.14	57.53
	St, Sn	37.69	45.34	72.16	71.94	37.17	44.81	55.25	56.42
	Wk, Un	40.75	47.49	68.56	71.41	37.64	43.81	53.58	53.92
	St, Un	38.03	46.57	65.36	65.21	36.25	42.02	52.86	52.92
J_{St}	Wk, Sn	41.00	47.55	73.75	75.69	38.51	44.92	56.13	57.75
	St, Sn	37.69	50.07	79.12	81.31	37.17	46.22	59.20	61.82
	Wk, Un	40.75	45.85	66.30	68.52	37.64	43.62	54.92	55.31
	St, Un	38.03	47.70	69.73	71.14	36.25	43.98	55.81	57.31
$J^{0.1}_{\mathrm{Wk}\to\mathrm{St}}$	Wk, Sn	41.00	–	73.70	–	–	–	–	–
	St, Sn	37.69	–	73.80	–	–	–	–	–
	Wk, Un	40.75	–	64.45	–	–	–	–	–
	St, Un	38.03	–	62.37	–	–	–	–	–
$J^{1.0}_{\mathrm{Wk}\to\mathrm{St}}$	Wk, Sn	41.00	–	78.22	–	–	–	–	–
	St, Sn	37.69	–	75.45	–	–	–	–	–
	Wk, Un	40.75	–	66.83	–	–	–	–	–
	St, Un	38.03	–	66.08	–	–	–	–	–

Table 4: *Judge's Consistent Accuracy.* Left block: DeepScaler; right block: MMLU-Pro. *Train* indicates whether the judge is trained on Weak data (J_{Wk}), Strong data (J_{St}), or via continual weak-to-strong training ($J^{\beta}_{\mathrm{Wk}\to\mathrm{St}}$). *Eval* indicates the type of evaluation split defined by the source of responses among Weak (Wk) or Strong (St) and whether questions are Seen (Sn) or Unseen (Un). Within each dataset, columns correspond to the judge-training configurations: Zero-Shot, SFT, DPO, and SFT+DPO. For both datasets, backbone (zero-shot) values are repeated across all blocks to facilitate direct comparison across judge-training strategies.