Rebuttal of Easy-to-Hard Generalization: Scalable Alignment Beyond Human Supervision

Anonymous Authors[1]

Methodology

...

0.1. Evaluators

Similarly, we consider the following evaluator models that can be trained either on the easy tasks only, or on the full dataset. Notably, unlike final-answer rewards, reward models trained on easy tasks can be applied to evaluate solutions to hard problems.

Final-Answer Reward is a symbolic reward that provides a binary reward based on the accuracy of the model's final answer. The matching is performed after normalization[1].

Outcome Reward Model (ORM) is trained on the Final-Answer rewards. Following Cobbe et al. (2021); Uesato et al. (2022); Lightman et al. (2023), we train the reward head to predict on every token whether the solution is correct, in a similar sense to a value model (Yu et al., 2023a). At inference time, we use the ORM's prediction at the final token as the reward of the solution.

Process Reward Model (PRM) is trained to predict whether each step (delimited by newlines) in the chain-of-thought reasoning path is correct. The labels are usually labeled by humans (Uesato et al., 2022; Lightman et al., 2023) or estimated with rollouts (Silver et al., 2016; Wang et al., 2023).

Outcome & Process Reward Model (OPRM) Building on the distinct advantages of ORMs and PRMs, we introduce the *Outcome & Process Reward Model (OPRM)*, which harnesses the complementary strengths of both. OPRM is trained on the mixed data of ORMs and PRMs. Specifically, OPRM is designed to evaluate the correctness of each intermediate reasoning step, akin to PRMs, while also assessing the overall solution's accuracy at the final answer stage, mirroring the functionality of ORMs.

0.2. Optimizing Generators Against Evaluators

Finally, given a generator model (i.e., policy model) and a evaluator model (i.e., reward model; RM), we optimize the generator against the evaluator using either re-ranking or reinforcement learning.

Best-of-n (BoN), also known as rejection sampling, is a reranking approach that sample multiple solutions from the generator and selects one with the highest RM score.

Weighted Voting is similar to majority voting or self-consistency (Wang et al., 2022), but weights each solution according to its RM score (Uesato et al., 2022).

Reinforcement Learning (RL) We consider three online/offline RL variants, Reinforced Self-Training (ReST) (Gulcehre et al., 2023; Singh et al., 2023), Direct Policy Optimization (DPO) (Rafailov et al., 2023), and Proximal Policy Optimization (PPO) (Schulman et al., 2017). Due to the space limit, please find their detailed description in Appendix A.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]https://github.com/openai/prm800k/blob/main/prm800k/grading/grader.py

Table 1. Easy-to-hard generalization of generators. We compare generator performance under various decoding settings. PRM800K and METAMATH indicate the SFT training data and ICL exemplars. Evaluations are performed on the same MATH500 test set.

		PRM800K			METAMATH		
		GREEDY	MAJ@16	MAJ@256	GREEDY	MAJ@16	MAJ@256
LLEMMA-7B	FULL ICL	12.8	15.6	20.8	16.4	18.4	25.6
	HARD ICL	12.6	18.0	27.0	16.6	19.0	27.0
	EASY-TO-HARD ICL	14.0	17.6	24.4	14.2	17.4	26.8
	FULL SFT	20.6	32.0	36.2	31.4	40.2	41.6
	EASY-TO-HARD SFT	19.8	31.6	36.0	30.0	38.6	42.4
LLEMMA-34B	FULL ICL	18.6	23.6	36.0	20.6	28.8	39.2
	HARD ICL	15.8	21.4	34.2	21.8	26.4	38.6
	EASY-TO-HARD ICL	18.2	25.2	36.8	19.8	26.8	37.2
	FULL SFT	25.6	41.8	46.4	35.4	44.2	45.6
	EASY-TO-HARD SFT	24.8	40.8	46.0	32.2	42.6	43.4

0.3. Evaluation Metrics

In this study, we have chosen not to establish terms analogous to the weak-to-strong performance gap recovery (PGR) as discussed in Burns et al. (2023) or the easy-to-hard supervision gap recovery (SGR) highlighted by Hase et al. (2024). This decision is based on our observations that sometimes, models trained exclusively on simpler tasks—particularly when employing RL training—can outperform those trained across the entire spectrum of problem difficulties. Therefore, we mainly focus on the absolute and relative performance of generators (optionally optimized by the evaluator) on the MATH500 test set (Lightman et al., 2023).

0.4. Implementation Details

Base Language Model Llemma is a large language model for mathematics (Azerbayev et al., 2023), which is continue pre-trained from Code Llama (Roziere et al., 2023) / LlaMA-2 (Touvron et al., 2023). We use both 7b and 34b variants in our experiments.

SFT / RL / Reward Model We fine-tune all models in full fine-tuning with frozen input-output embedding layers and normalization layers. RMs are initialized from the base model, and have an added scalar head to output the reward. In PPO training, we initialize the value model from the reward model.

Hyper-parameters Due to the space limit, our training hyper-parameters can be found in Appendix. B.

Main Results

0.5. Easy-to-Hard Generalization of Generators

In Table 1, we compare the easy-to-hard generalization performance of the generators under various decoding settings:

Supervised Fine-Tuning (SFT) outperforms In-Context Learning (ICL): This is consistent with prior work (Stiennon et al., 2020; Ouyang et al., 2022; Uesato et al., 2022). We also find that the performance of ICL has larger variance than SFT with respect to data ordering (or random seeds) (Dodge et al., 2020; Zhao et al., 2021).

SFT data quality impacts easy-to-hard generalization: PRM800K data is generated by a base (unaligned) GPT-4 model through few-shot prompting and is thus of lower quality than well-aligned ChatGPT-generated MetaMATH data. We find that only MetaMath-trained models have certain easy-to-hard gaps (e.g., 16.6 v.s. 14.2 in MetaMath-7b-ICL), while such gaps in PRM800K-trained models are very small (less than 1%), or even inverted in the ICL setting. We hypothesize that low-quality SFT data may only teach the model the format of the task (Sanh et al., 2021; Wei et al., 2021; Wang et al., 2022), while high-quality (imitation) SFT data can teach the model the principles of solving the task (Sun et al., 2023b; Gudibande et al., 2023). Nevertheless, the strongest performance is achieved by full SFT on the high-quality MetaMath data (35.4), showing an unignorable difference, with a gap of up to 3.2, compared to its easy-to-hard SFT counterpart (32.2).

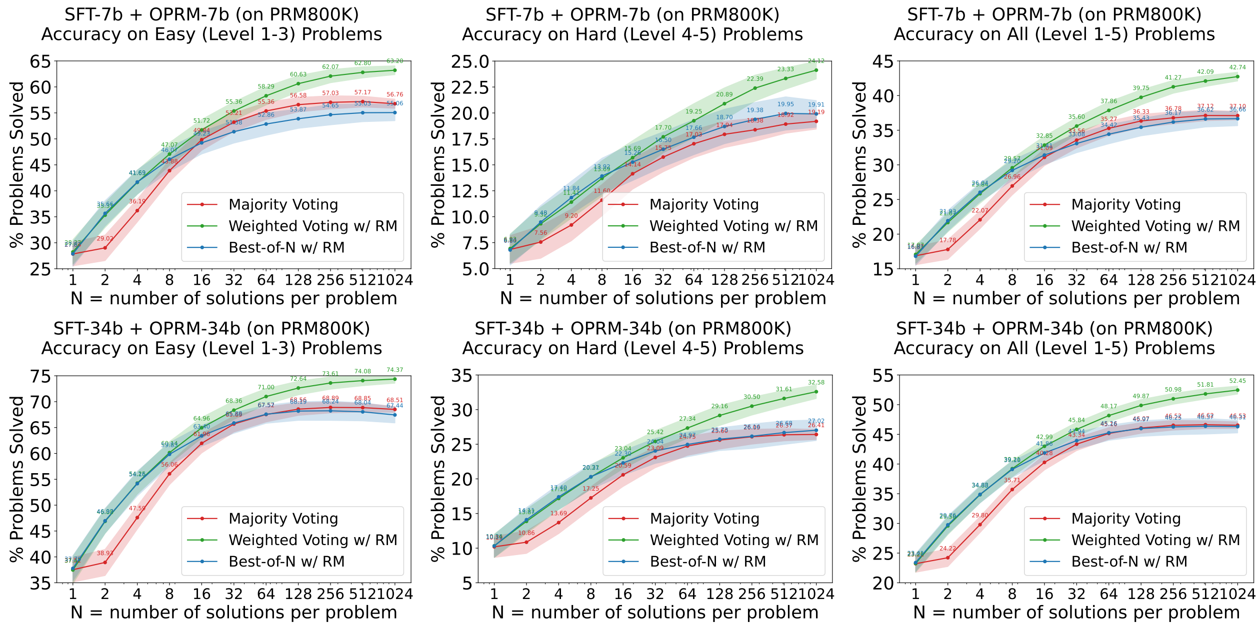

Figure 1. Easy-to-hard generalization of 7b (upper) and 34b (lower) evaluators. Both SFTs and RMs are trained on the easy data. We found that PRMs trained on easy tasks can significantly improve the re-ranking (i.e., weighted voting) performance on hard tasks.

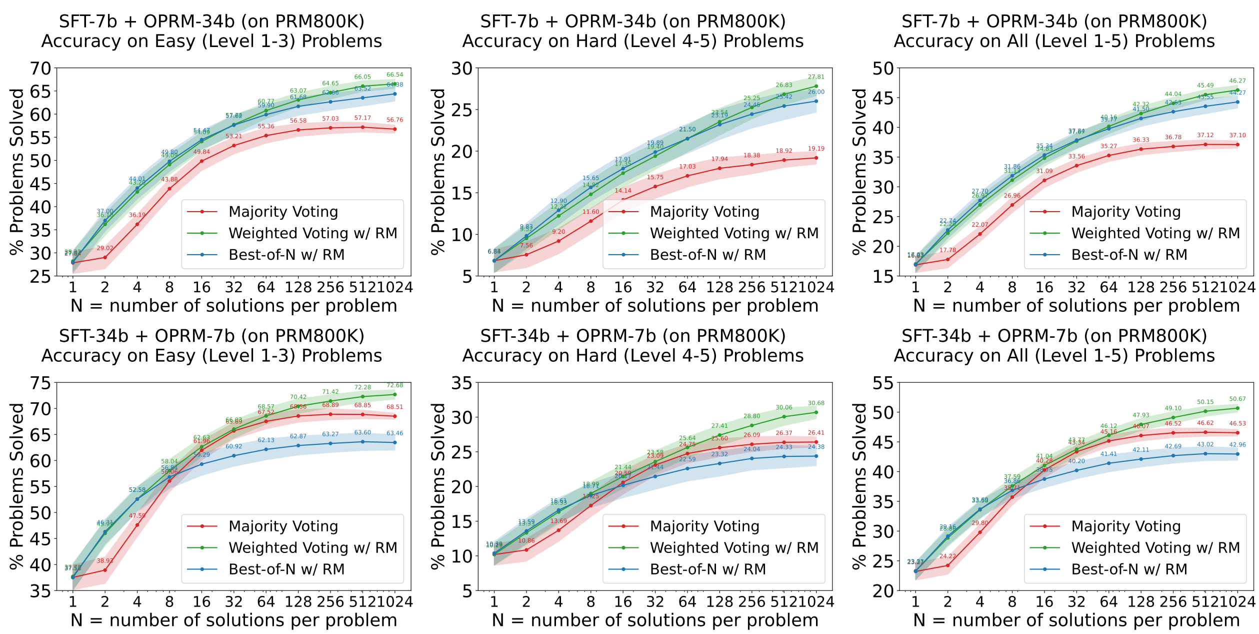

Figure 2. Easy-to-hard generalization of evaluators applied to generators of different sizes. We evaluated 7b generator + 34b evaluator (upper) and 34b generator + 7b evaluator (lower). Both SFTs and RMs are trained on the easy data.

0.6. Easy-to-Hard Generalization of Evaluators

The primary metric we use to assess the effectiveness of our process reward model is not the average accuracy of verifying each step in a solution but rather the overall performance achieved through re-ranking methods (See discussion in Sec. 0.3). We first use re-ranking to evaluate the easy-to-hard generalization performance of evaluators.

Table 2. Comparing reinforcement learning (RL) approaches for easy-to-hard generalization. All methods are of 7b size and evaluated with greedy decoding. † indicates the model is trained with additional final-answer labels on hard tasks (similar to Singh et al. (2023)), which is not strictly a easy-to-hard generalization setup.

	RL Data	Reward		Accuracy		
		Final-Answer	Process RM	Easy (Level 1-3)	Hard (Level 4-5)	All
(SFT / PRM trained on level 1-3 of PRM800K)						
SFT				28.2	12.2	19.8
ReST-EM	Easy	Easy	×	33.2	12.6	22.4
ReST-EM	Hard	Hard	×	31.9	8.0	19.4
ReST-EM†	All	All	×	35.7	8.8	21.6
Iterative DPO	Easy	Easy	√	42.0	12.2	26.4
Iterative DPO†	All	All	√	38.2	11.5	24.2
PPO	Easy	Easy	×	42.0	14.1	27.4
PPO	Hard	Hard	×	34.0	9.2	21.0
PPO†	All	All	×	42.0	10.7	25.6
PPO	All	Easy	√	**45.4**	**14.9**	**29.4**
PPO	All	Easy	√	**45.4**	**14.9**	**29.4**
(SFT / PRM trained on level 1-5 of MetaMath / Math-Shepherd)						
Llemma-based SFT SoTA (Ours)				51.7	13.7	31.4
Previous RL SoTA (Wang et al., 2023)				-	-	33.0
(SFT / PRM trained on level 1-3 of MetaMath / Math-Shepherd)						
SFT				44.1	14.9	28.8
ReST-EM	Easy	Easy	×	50.4	14.5	31.6
Iterative DPO	Easy	Easy	√	**53.8**	**16.0**	**34.0**
Iterative DPO†	All	All	√	47.9	12.2	29.2
PPO	Easy	Easy	×	50.8	15.3	32.2
PPO†	All	All	×	50.8	13.4	31.2
PPO	All	Easy	√	**53.8**	**16.0**	**34.0**

0.6.1. Re-ranking

We consider two re-ranking strategies: Best-of-n (or rejection sampling) and Weighted Voting. In our easy-to-hard generalization setting, both SFT models and Reward Models (RMs) are trained on easier tasks (levels 1-3), but evaluated on all difficulty levels (1-5). We compare the performance between majority voting (SFT only) and re-ranking (SFT + OPRM) on the PRM800K dataset in Figure 1-2, and the performance of different reward models (PRMs, ORMs, & OPRMs) on the PRM800K dataset in Figure 5-6. Specifically, we use `min` as the reward aggregation function for best-of-n and `prod` for weighted voting[2]. The figures illustrate the performance of different decoding strategies or reward models under the same number of sampled solutions per problem. We have the following findings:

OPRMs outperforms ORMs and PRMs This confirms our hypothesis that Process Reward Models (PRMs) and Outcome Reward Models (ORMs) capture different aspects of task-solving processes. By integrating the strengths of both PRMs and ORMs, Outcome & Process Reward Models (OPRMs) demonstrate superior performance. However, follow-up experiments conducted on the MetaMath/Math-Shepherd datasets do not demonstrate significant improvements from incorporating additional ORM training examples. This lack of enhancement may be attributed to the fact that Math-Shepherd is already generated from final-answer reward generation. This suggests that there remains a substantial difference between process rewards labeled by humans (e.g., PRM800K) and those generated automatically (e.g., Math-Shepherd).

Weighted voting outshines Best-of-n This finding diverges from past research where minimal performance differences were observed between weighted voting and Best-of-n (Lightman et al., 2023; Uesato et al., 2022). Our hypothesis is that this discrepancy arises from our specific experiment, which involves training a less powerful base model (Llemma; Azerbayev et al. 2023) on more difficult tasks (MATH; Hendrycks et al. 2021). This setup might diminish the effectiveness of the reward model, potentially leading to an over-optimization of rewards (Gao et al., 2023). Given these insights, weighted voting is preferred as the primary re-ranking method for further discussions. Nevertheless, Best-of-n still achieves competitive performance to majority voting when producing only one full solution. In Figure 2, we also find that the 34b evaluator can

[2] See more detailed analysis of reward aggregation functions in Appendix. G.

significantly improve the 7b generator, while the 7b evaluator can still improve the performance of the 34b generator.

Greater effectiveness of re-ranking on harder tasks: Weighted voting not only consistently surpasses majority voting but also shows a more pronounced advantage on harder tasks. This observation leads to the conclusion that *evaluators demonstrate better easy-to-hard generalization capabilities in comparison to generators*. This motivates us to explore RL approaches that optimize the generator against the evaluator to further improve the performance of easy-to-hard generation.

0.6.2. Reinforcement Learning (RL)

Given the conclusion above, an important question arises: how can evaluators once again assist generators in achieving enhanced easy-to-hard generalization capabilities? We further investigate the enhancement of policy models through RL, utilizing easy-to-hard evaluators as reward models. Similar to re-ranking, SFT and PRM are only trained on easy data. For a fair comparison between PRM800K and MetaMath, we only use vanilla PRMs in the RL training. All the RL methods use the validation accuracy for selecting the best checkpoint[3]. Our comparison spans offline (ReST & DPO) and online (PPO) RL algorithms under various training conditions:

Easy Questions & Easy Final Answers: The SFT model samples from easy questions and receives the corresponding Final-Answer and optional PRM rewards.

All Questions & Easy Final Answers: This assumes access to a range of easy and hard problems for RL training, with rewards for hard tasks solely provided by the easy-to-hard evaluator.

All Questions & All Final Answers: This setting uses all data with the corresponding final answers, which is similar to Singh et al. (2023), but not strictly a easy-to-hard generalization setup.

Based on the results reported in Table 2, we have the following findings:

DPO and PPO excel over ReST: Among the RL algorithms trained on the PRM800K dataset, PPO emerges as the most effective, significantly surpassing both ReST and DPO. On the MetaMATH dataset, PPO and DPO achieve top performance, while ReST shows only marginal improvements over the SFT baseline. The comparative analysis between DPO and PPO across the PRM800K and MetaMATH datasets indicates that while DPO's efficacy is on par with PPO given a high-quality SFT model as initialization, PPO's effectiveness is less contingent on the quality of the underlying SFT model (Ouyang et al., 2022; Rafailov et al., 2023).

PRM rewards are more beneficial than Final-Answer rewards for hard tasks: Notably, models trained with PRM rewards with human supervision on the easy tasks (achieving a top performance of 34.0) outperform the previous state-of-the-art model trained across all task levels (33.0). This highlights the effectiveness of leveraging easy-to-hard evaluations to improve generator performance across varying task difficulties.

[3]This includes stopping iterations in ReST-EM and iterative DPO, and stopping online steps in PPO.

A. Reinforcement Learning Algorithms

Reinforced Self-Training (ReST) is an offline RL algorithm, which alternates between generating samples from the policy, which are then used to improve the LLM policy with RM-weighted SFT (Gulcehre et al., 2023; Singh et al., 2023). Its variants include expert iteration (Anthony et al., 2017) and rejection sampling fine-tuning (Touvron et al., 2023; Yuan et al., 2023).

Direct Policy Optimization (DPO) is a class of offline RL algorithms (Rafailov et al., 2023) that consider both positive and negative gradient updates. It fine-tunes the policy model on a preference dataset consisting of paired positive and negative samples. The variants include NLHF (Munos et al., 2023), IPO (Azar et al., 2023), and SLiC (Zhao et al., 2022; 2023). Recent work shows that iteratively applying DPO leads to improved performance (Xu et al., 2023).

Proximal Policy Optimization (PPO) is an online RL algorithm which samples from the policy during fine-tuning (Schulman et al., 2017). It is widely used in RLHF (Stiennon et al., 2020; Bai et al., 2022a; Ouyang et al., 2022) and RLAIF (Bai et al., 2022b; Sun et al., 2023a).

B. Hyper-parameters

B.1. Supervised Fine-Tuning & Reward Modeling

For the PRM800K dataset (Lightman et al., 2023), the SFT model is trained using steps that are labeled as correct. For the MetaMath dataset (Yu et al., 2023b), given that the original dataset can contain upwards of ten solutions for the same question, potentially leading to over-fitting, we implement a filtering process. This process ensures that, during any given epoch, no more than three solutions per question are retained, thereby mitigating the risk of over-fitting.

The PRMs are trained on the corresponding released dataset (Lightman et al., 2023; Wang et al., 2023). For generating solutions to train ORMs, we sample 32 solutions for each question from the language model using top-K sampling with K=20 and temperature of 0.7. We also ensure that the ratio between positive and negative samples for each question is between 1:3 to 3:1.

See Table 3 for a list of training hyper-parameters used in the training jobs. We use full fine-tuning for all SFT/RM training.

Table 3. Hyper-parameters in our SFT/RM training jobs

		PRM800K				MetaMath	
		SFT	PRM	ORM	OPRM	SFT	PRM
Llemma-7b	Learning Rate	2e-5	2e-5	2e-5	2e-5	8e-6	2e-5
	Epochs	3	2	2	2	3	2
	Batch Size	128	128	128	128	128	128
	Max Seq Len	768	768	1024	1024	1024	768
	Dtype	BF16	BF16	BF16	BF16	FP32	BF16
Llemma-34b	Learning Rate	1e-5	1e-5	1e-5	1e-5	5e-6	-
	Epochs	3	2	2	2	3	-
	Batch Size	128	128	128	128	128	-
	Max Seq Len	768	768	1024	1024	768	-
	Dtype	BF16	BF16	BF16	BF16	FP32	-

B.2. Re-Ranking

For majority voting, weighted voting, and best-of-n, we sample from the language model using top-K sampling with K=20 and temperature of 0.7. At test time, we use the ORM's prediction at the final token as the overall score for the solution, and use the PRM's prediction at each intermediate step (denoted by the new line symbol) and the final token as the process reward scores.

B.3. Reinforcement Learning

We use full fine-tuning during the RL stage.

ReST-EM Following Singh et al. (2023), we sample 32 solutions for each question from the language model using top-K sampling with K=40. We also used a cut-off threshold of 10 for the maximum number of solutions per problem (Zelikman et al., 2022; Singh et al., 2023). We performed iterative ReST training for two epochs, and observed performance degeneration starting from the third epoch. For PRM800K, we used a temperature of 1.0, while for MetaMath, we used a temperature of 1.2. The rest training hyper-parameters are the same as in SFT training.

Iterative DPO We sample 8 solutions for each question from the language model using top-K sampling with K=20 and temperature of 1.0. We use the process reward model to assign a score between 0 and 1 to each solution, and use final-answer reward to assign an additional 0/1 score to each solution. A preference training pair is constructed only when the score difference between positive and negative solutions is greater than 1.0. We used a cut-off threshold of 3 for the maximum number of preference pairs per problem.

For all DPO training (Rafailov et al., 2023), we used a learning rate of 2×10^{-6}, a batch size of 64, and a DPO training epoch of 1. We set $\beta = 0.1$ for all DPO experiments, and performed at most 5 DPO iterations (i.e., sampling new solutions and performing one DPO epoch).

PPO We follow Dubois et al. (2023) on the implementation of the PPO algorithm, which is a variant of (Ouyang et al., 2022)[4]. Specifically, we normalize the advantage across the entire batch of rollouts obtained for each PPO step and initialize the value model from the reward model.

We clipped the gradient by its Euclidean norm at a limit of 1. Our training spanned 500 PPO steps on the RL data (MATH questions except MATH500 and our 500 validation questions). For generalized advantage estimation (GAE; Schulman et al. (2015)), both λ and γ were set at 1.

For PRM800K, we used a batch size of 512 for each PPO step. This comprised 8 epochs of gradient steps, each having 64 rollouts. We applied a peak learning rate of 2×10^{-5} with cosine decay. We opted for a constant KL regularizer coefficient of 0.01, and a sampling temperature of 0.7.

For MetaMath/Math-Shepherd, we used a batch size of 512 for each PPO step. This comprised 2 epochs of gradient steps, each having 256 rollouts. We applied a peak learning rate of 5×10^{-6} with cosine decay. We opted for a constant KL regularizer coefficient of 0.002, and a sampling temperature of 1.2.

C. Re-ranking Results on MetaMath

Similar to Sec. 0.6.1, we assess the effectiveness of process reward models on the MetaMath/Math-Shepherd dataset (Yu et al., 2023b; Wang et al., 2023). From Figure 3, we can see that PRMs are also more effective on harder tasks when trained on MetaMath/Math-Shepherd.

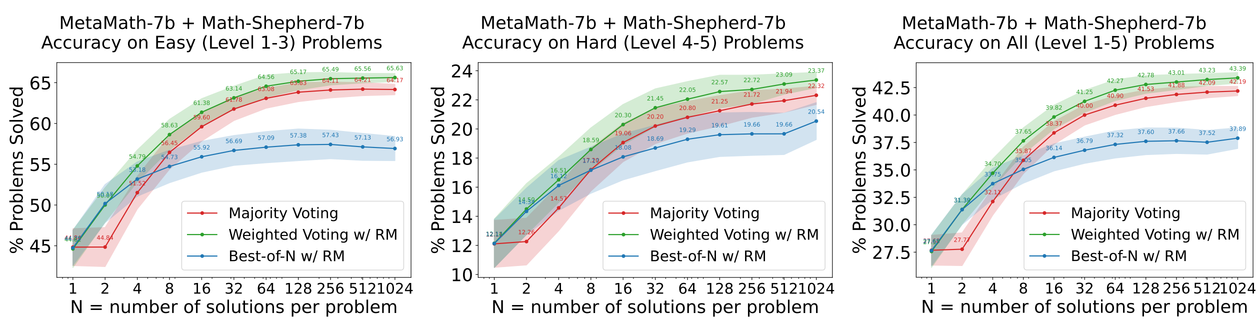

Figure 3. Easy-to-hard generalization of evaluators on MetaMath. Both SFTs and RMs are trained on the easy data.

[4] https://github.com/openai/lm-human-preferences

D. Qualitative Comparison

QUESTION	The value of y varies inversely as $\sqrt{x}$ and when $x = 24$, $y = 15$. What is x when $y = 3$?
METAMATH-SFT	If y varies inversely as $\sqrt{x}$, then $y = \frac{k}{\sqrt{x}}$ for some constant k. Plugging in the given values, we have $15 = \frac{k}{\sqrt{24}}$, or $k = \frac{15\sqrt{24}}{24} = \frac{15\sqrt{6}}{4}$. Now we can find x when $y = 3$ by plugging in these values into the equation $y = \frac{k}{\sqrt{x}}$. We have $3 = \frac{\frac{15\sqrt{6}}{4}}{\sqrt{x}}$, or $3 = \frac{15\sqrt{6}}{4\sqrt{x}}$. Cross-multiplying, we get $3 \cdot 4\sqrt{x} = 15\sqrt{6}$, or $12\sqrt{x} = 15\sqrt{6}$. Dividing both sides by 12, we find $\sqrt{x} = \frac{15\sqrt{6}}{12\sqrt{6}} = \frac{5}{2}$. Squaring both sides, we get $x = \left(\frac{5}{2}\right)^2 = \boxed{6.25}$.
METAMATH-PPO	If y varies inversely as $\sqrt{x}$, then we can write the equation as $y = \frac{k}{\sqrt{x}}$ for some constant k. We are given that when $x = 24$, $y = 15$, so we can substitute these values into the equation to solve for k: $15 = \frac{k}{\sqrt{24}}$ Multiplying both sides by $\sqrt{24}$ gives $15\sqrt{24} = k$ Simplifying the left side gives $15 \cdot 2\sqrt{6} = k$, which further simplifies to $30\sqrt{6} = k$ Now we can use this value of k in the equation to find x when $y = 3$: $3 = \frac{30\sqrt{6}}{\sqrt{x}}$ Multiplying both sides by $\sqrt{x}$ gives $3\sqrt{x} = 30\sqrt{6}$ Dividing both sides by 3 gives $\sqrt{x} = 10\sqrt{6}$ Squaring both sides gives $x = (10\sqrt{6})^2 = 100(6) = \boxed{600}$

Table 4. Qualitative comparison between MetaMath-7b-SFT and MetaMath-7b-PPO, as well as the first error detected in MetaMath-SFT greedy decoding by Math-Shepherd. Examples are cherry-picked for better illustration.

E. Performance of DPO Iterations

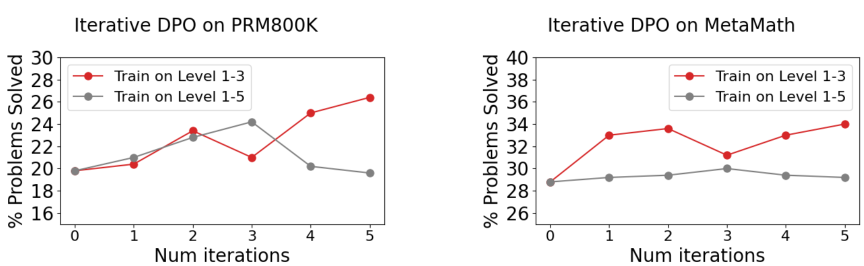

Figure 4. Test performance as a function of DPO iterations.

F. Re-ranking Performance Analysis of PRMs, ORMs & OPRMs

We compare the re-ranking performance of Process Reward Models (PRMs), Outcome Reward Models (ORMs), and our proposed Outcome & Process Reward Models (OPRMs). Figure 5 shows the results on 7b models and Figure 6 is on 34b models. We find that in our setting of Llemma (Azerbayev et al., 2023) + MATH (Hendrycks et al., 2021), PRMs and ORMs perform similarly, with PRMs slightly outperforming ORMs on hard tasks. But the OPRMs that trained on the mixed data of PRMs and ORMs significantly outperforms both of them.

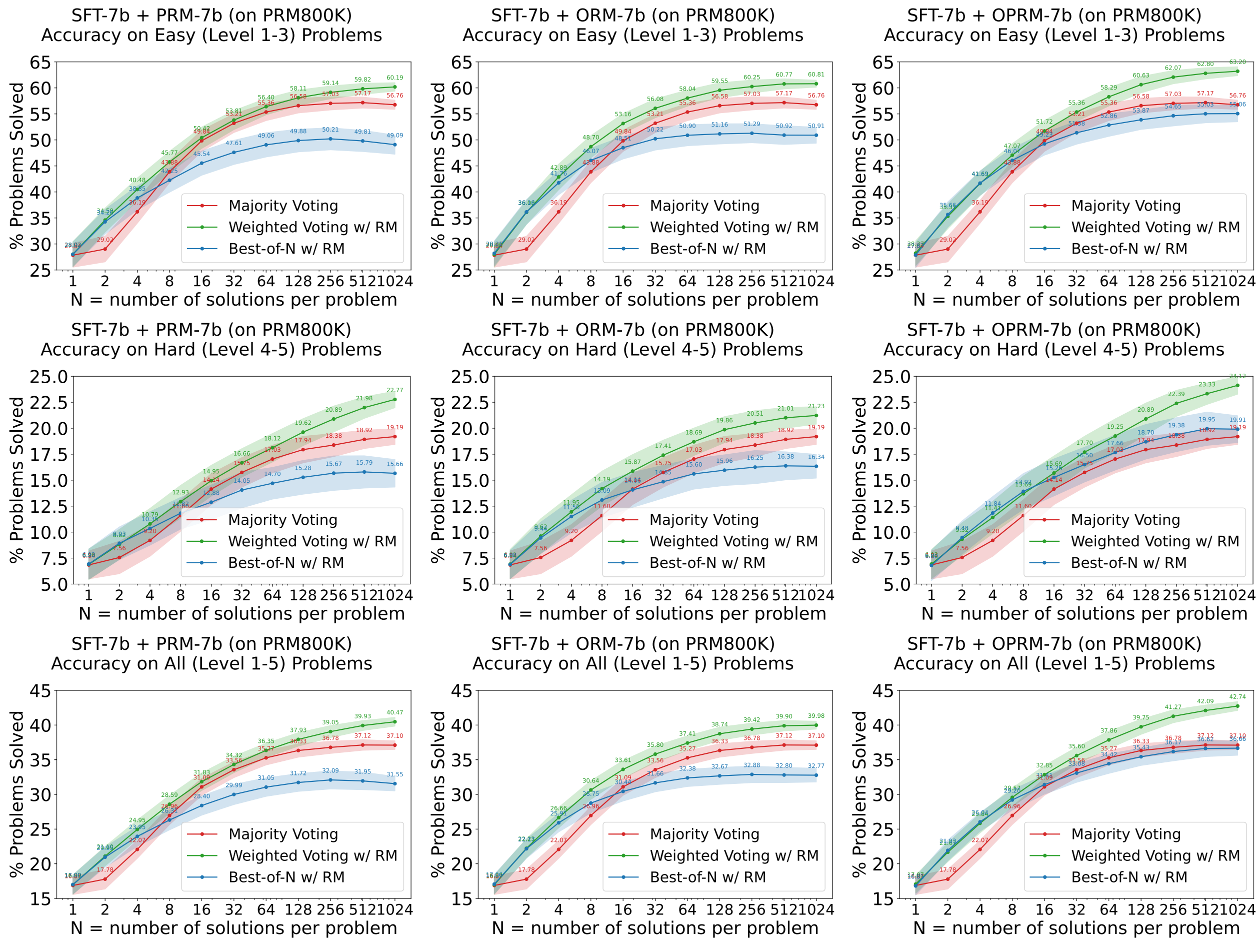

Figure 5. Comparing process reward models (PRMs, left), outcome reward models (ORMs, middle), and outcome & process reward models (OPRMs, right) on 7b models trained on the PRM800K dataset. Both SFTs and RMs are trained on the easy data.

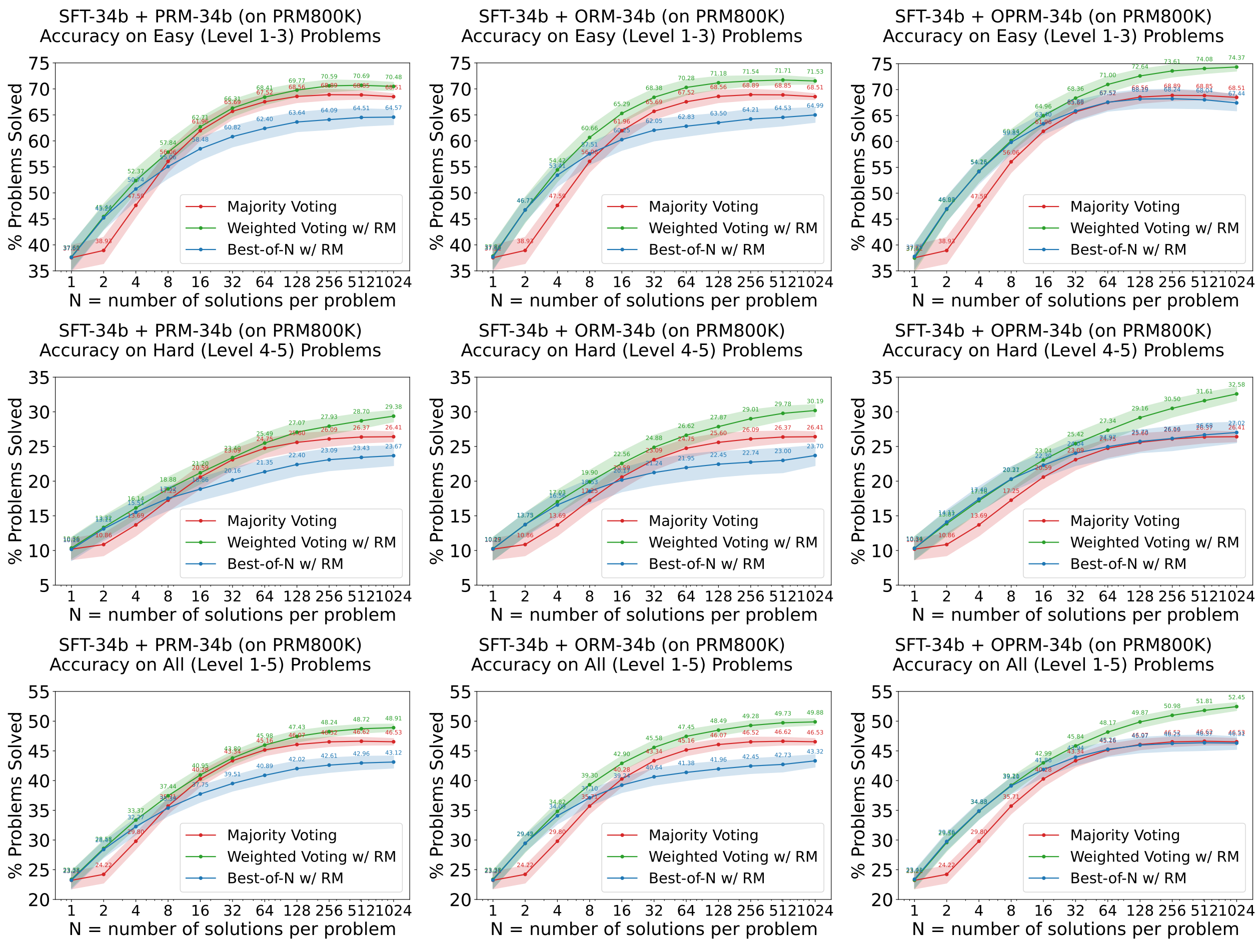

Figure 6. Comparing process reward models (PRMs, left), outcome reward models (ORMs, middle), and outcome & process reward models (OPRMs, right) on 34b models trained on the PRM800K dataset. Both SFTs and RMs are trained on the easy data.

G. Analysis of Aggregation Functions in PRMs & OPRMs

We explored different methods to consolidate step-wise prediction scores into a single score value, a process we describe as employing an aggregation function, during the use of the evaluator. Lightman et al. (2023) report comparable performance when using `min` (minimum) and `prod` (product) as the aggregation function to reduce multiple scores into a single solution-level score. Note that when training PRMs on PRM800K (Lightman et al., 2023), we have already considered neutral steps to be positive as training labels.

Following Wang et al. (2024), given $\{p_1, p_2, \ldots, p_n\}$ as a list of predicted correctness probability of each step (including the final answer), we considered the following aggregation functions:

$$\texttt{min} = \min\{p_1, p_2, \ldots, p_n\} \tag{1}$$

$$\texttt{max} = \max\{p_1, p_2, \ldots, p_n\} \tag{2}$$

$$\texttt{prod} = \prod_i p_i \tag{3}$$

$$\texttt{mean} = \frac{\sum_i p_i}{n} \tag{4}$$

$$\texttt{mean\_logit} = \sigma\left(\frac{\sum_i \log \frac{p_i}{1-p_i}}{n}\right) \tag{5}$$

$$\texttt{mean\_odd} = \mathrm{ReLU}\left(\frac{\sum_i \frac{p_i}{1-p_i}}{n}\right) \tag{6}$$

$$\texttt{last} = p_n \tag{7}$$

In Figure 7-9, we perform analysis of aggregation functions on PRM800K and Math-Shepherd (from MetaMath) datasets with weighted voting and best-of-n decoding and PRMs or OPRMs. In general, we find `prod` works universally well in weighted voting and `min` works well in best-of-n. So we adopt these two strategies in our main experiments.

One interesting finding is that for reward models trained on the human annotated process reward (e.g., PRM800K), the `last` strategy does not perform very well, but `last` works much better on OPRMs and pseudo PRMs (e.g., Math-Shepherd). This could partially explain why OPRMs does not further improve the performance on the Math-Shepherd dataset.

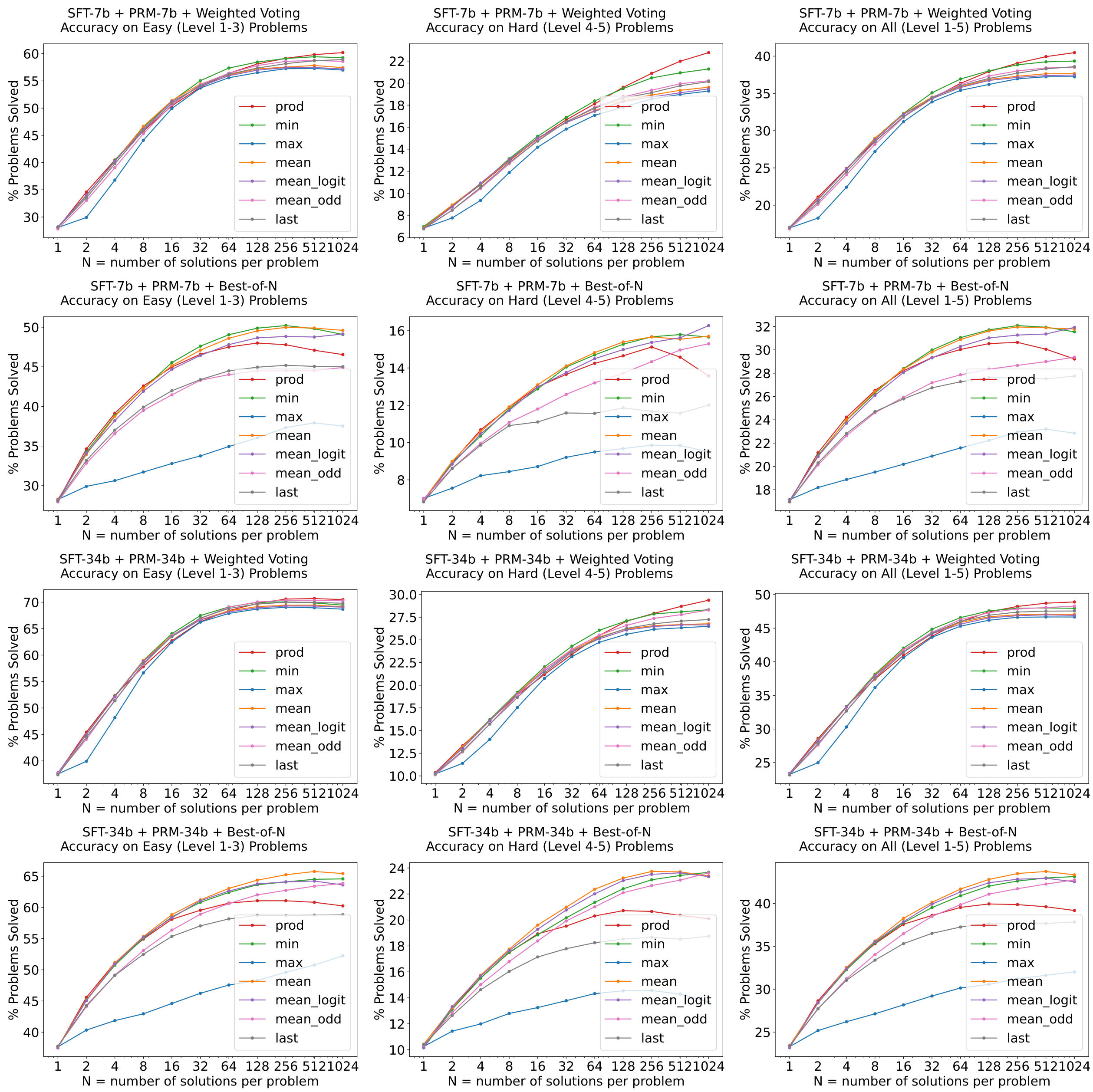

Figure 7. Analysis of aggregation functions in process reward models (PRMs) on the PRM800K dataset with Weighted Voting and Best-of-N. Both SFTs and RMs are trained on the easy data.

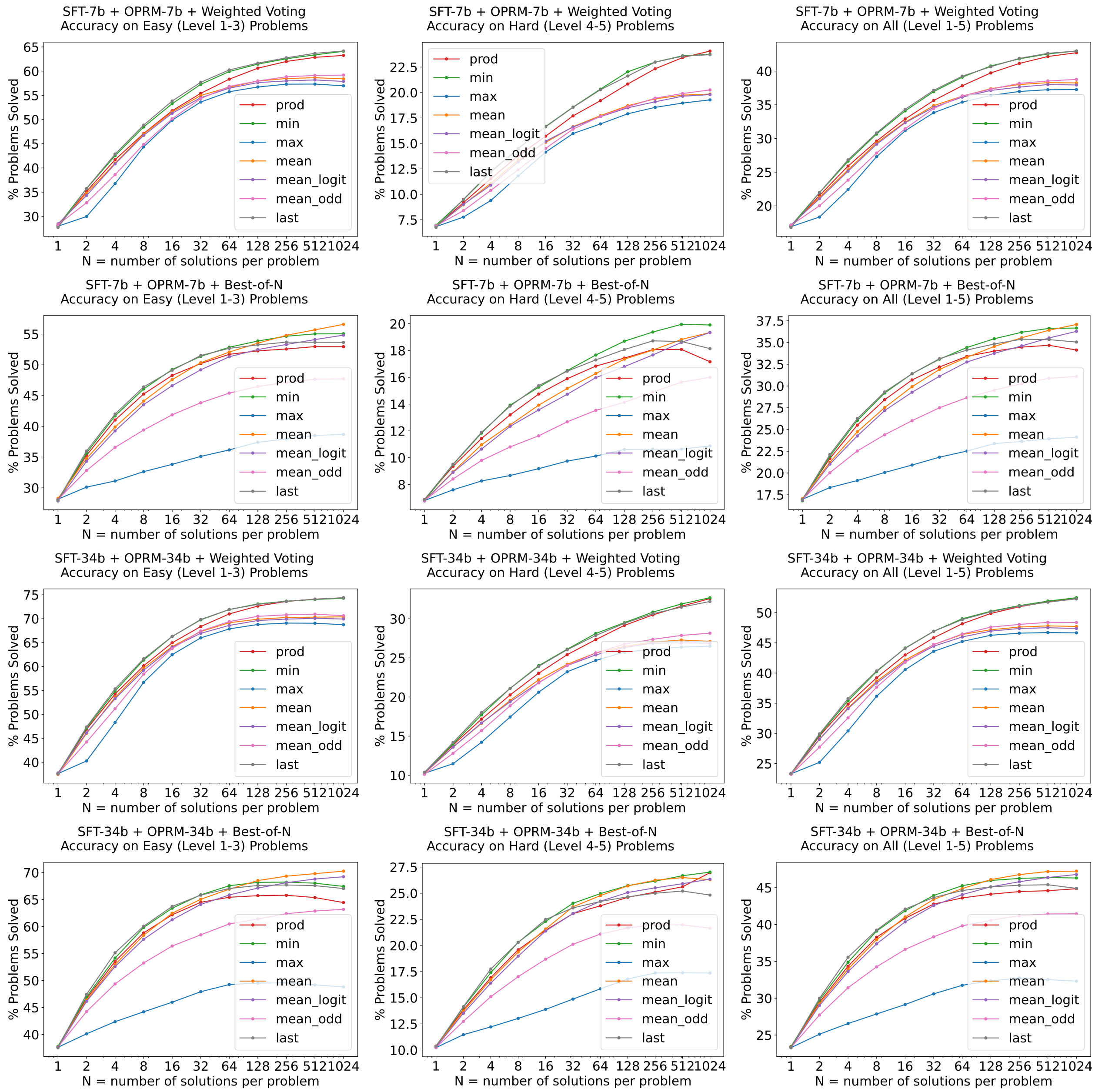

Figure 8. Analysis of aggregation functions in outcome & process reward models (OPRMs) on the PRM800K dataset with Weighted Voting and Best-of-N. Both SFTs and RMs are trained on the easy data.

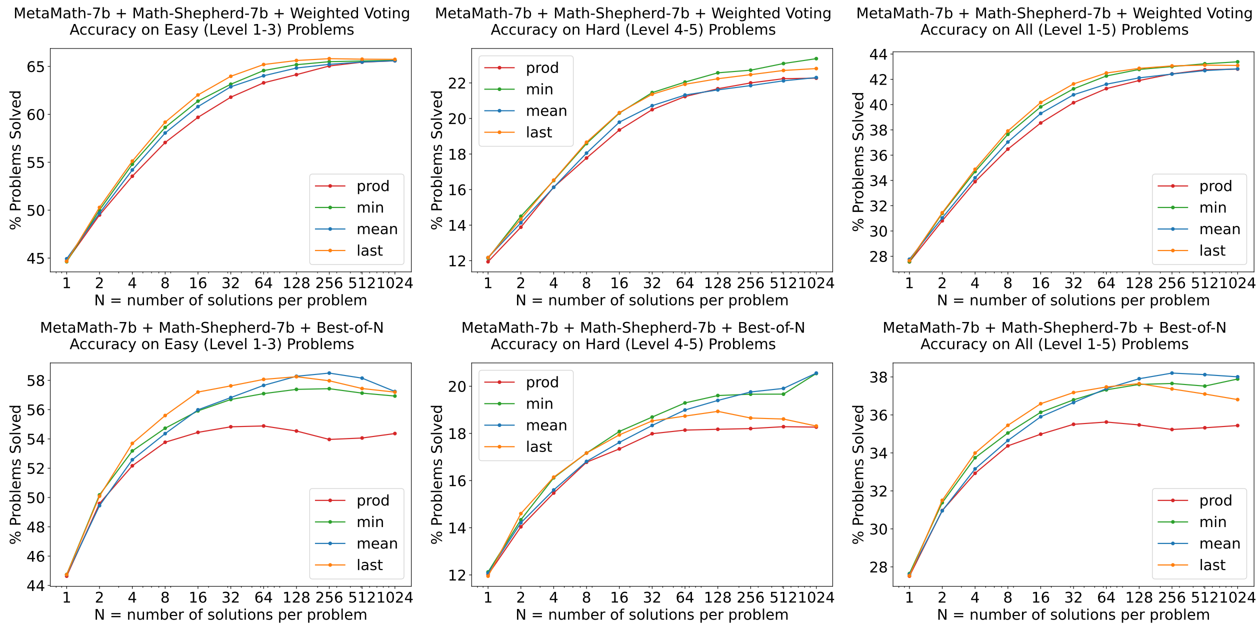

Figure 9. Analysis of aggregation functions in psuedo process reward models (PRMs) on the Math-Shepherd (from MetaMath) dataset with Weighted Voting and Best-of-N. Both SFTs and RMs are trained on the easy data.

References

Anthony, T., Tian, Z., and Barber, D. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

Azar, M. G., Rowland, M., Piot, B., Guo, D., Calandriello, D., Valko, M., and Munos, R. A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036*, 2023.

Azerbayev, Z., Schoelkopf, H., Paster, K., Santos, M. D., McAleer, S., Jiang, A. Q., Deng, J., Biderman, S., and Welleck, S. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*, 2023.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., Chen, C., Olsson, C., Olah, C., Hernandez, D., Drain, D., Ganguli, D., Li, D., Tran-Johnson, E., Perez, E., Kerr, J., Mueller, J., Ladish, J., Landau, J., Ndousse, K., Lukosuite, K., Lovitt, L., Sellitto, M., Elhage, N., Schiefer, N., Mercado, N., DasSarma, N., Lasenby, R., Larson, R., Ringer, S., Johnston, S., Kravec, S., Showk, S. E., Fort, S., Lanham, T., Telleen-Lawton, T., Conerly, T., Henighan, T., Hume, T., Bowman, S. R., Hatfield-Dodds, Z., Mann, B., Amodei, D., Joseph, N., McCandlish, S., Brown, T., and Kaplan, J. Constitutional ai: Harmlessness from ai feedback, 2022b.

Burns, C., Izmailov, P., Kirchner, J. H., Baker, B., Gao, L., Aschenbrenner, L., Chen, Y., Ecoffet, A., Joglekar, M., Leike, J., et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., and Smith, N. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020.

Dubois, Y., Li, X., Taori, R., Zhang, T., Gulrajani, I., Ba, J., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpacafarm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387*, 2023.

Gao, L., Schulman, J., and Hilton, J. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.

Gudibande, A., Wallace, E., Snell, C., Geng, X., Liu, H., Abbeel, P., Levine, S., and Song, D. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*, 2023.

Gulcehre, C., Paine, T. L., Srinivasan, S., Konyushkova, K., Weerts, L., Sharma, A., Siddhant, A., Ahern, A., Wang, M., Gu, C., et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

Hase, P., Bansal, M., Clark, P., and Wiegreffe, S. The unreasonable effectiveness of easy training data for hard tasks. *arXiv preprint arXiv:2401.06751*, 2024.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Munos, R., Valko, M., Calandriello, D., Azar, M. G., Rowland, M., Guo, Z. D., Tang, Y., Geist, M., Mesnard, T., Michi, A., et al. Nash learning from human feedback. *arXiv preprint arXiv:2312.00886*, 2023.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., et al. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2021.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529 (7587):484–489, 2016.

Singh, A., Co-Reyes, J. D., Agarwal, R., Anand, A., Patil, P., Liu, P. J., Harrison, J., Lee, J., Xu, K., Parisi, A., et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Sun, Z., Shen, Y., Zhang, H., Zhou, Q., Chen, Z., Cox, D., Yang, Y., and Gan, C. Salmon: Self-alignment with principle-following reward models. *arXiv preprint arXiv:2310.05910*, 2023a.

Sun, Z., Shen, Y., Zhou, Q., Zhang, H., Chen, Z., Cox, D., Yang, Y., and Gan, C. Principle-driven self-alignment of language models from scratch with minimal human supervision. *arXiv preprint arXiv:2305.03047*, 2023b.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Uesato, J., Kushman, N., Kumar, R., Song, F., Siegel, N., Wang, L., Creswell, A., Irving, G., and Higgins, I. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *CoRR, abs/2312.08935*, 2023.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

Wang, Z., Li, Y., Wu, Y., Luo, L., Hou, L., Yu, H., and Shang, J. Multi-step problem solving through a verifier: An empirical analysis on model-induced process supervision. *arXiv preprint arXiv:2402.02658*, 2024.

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2021.

Xu, J., Lee, A., Sukhbaatar, S., and Weston, J. Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682*, 2023.

Yu, F., Gao, A., and Wang, B. Outcome-supervised verifiers for planning in mathematical reasoning. *arXiv preprint arXiv:2311.09724*, 2023a.

Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J. T., Li, Z., Weller, A., and Liu, W. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023b.

Yuan, Z., Yuan, H., Li, C., Dong, G., Tan, C., and Zhou, C. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Zhao, Y., Khalman, M., Joshi, R., Narayan, S., Saleh, M., and Liu, P. J. Calibrating sequence likelihood improves conditional language generation. In *The Eleventh International Conference on Learning Representations*, 2022.

Zhao, Y., Joshi, R., Liu, T., Khalman, M., Saleh, M., and Liu, P. J. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

Zhao, Z., Wallace, E., Feng, S., Klein, D., and Singh, S. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pp. 12697–12706. PMLR, 2021.