Solving Math Word Problems with Process-based and Outcome-based Feedback

Jonathan Uesato* **Nate Kushman*** **Ramana Kumar*** **Francis Song†**

Noah Siegel **Lisa Wang** **Antonia Creswell** **Geoffery Irving** **Irina Higgins**

DeepMind

Abstract

Recent work has shown that prompting language models to generate reasoning steps improves performance on many reasoning tasks. When moving beyond prompting, this raises the question of how we should supervise the finetuning of such models: outcome-based approaches which supervise the final result, or process-based approaches which supervise the reasoning process itself? Differences between these approaches might naturally be expected not just in final-answer errors but also in reasoning errors, which can be difficult to detect and are problematic in many real-world domains such as education. We run the first comprehensive comparison between process- and outcome-based approaches trained on a natural language task, GSM8K. We find that pure outcome-based supervision produces similar final-answer error rates with less label supervision. However, for correct reasoning steps we find it necessary to use process-based supervision or supervision from learned reward models that emulate process-based feedback. In total, we improve the previous best results from 16.8% $\rightarrow$ 12.7% final-answer error and from 14.0% $\rightarrow$ 3.4% reasoning error among final-answer-correct solutions.

1 Introduction

Recent work has shown that asking language models to use step-by-step reasoning improves performance on reasoning tasks (Shwartz et al., 2020; Nakano et al., 2021; Cobbe et al., 2021; Wei et al., 2022; Kojima et al., 2022; Lewkowycz et al., 2022). While these works have primarily focused on prompting language models, prior work suggests that finetuning should outperform prompting alone (Stiennon et al., 2020; Perez et al., 2021; Ouyang et al., 2022). This raises the question of how best to supervise such models. Two natural approaches are outcome-based approaches, which supervise the final result, and process-based approaches, which supervise each step of the reasoning process, including the last step outputting the final result.

In this work, we conduct the first comprehensive comparison between process- and outcome-based approaches trained on a natural language task. For this, we use the recently proposed GSM8K dataset (Cobbe et al., 2021) of math word problems. In all cases, we generate a sequence of reasoning steps leading to the final answer, but vary whether or not supervision is provided only on the final answers (outcome-based) or on individual reasoning steps (process-based). For process-based approaches we consider supervision provided by both offline human-generated reasoning traces from the GSM8K dataset itself, as well as online human correctness annotations, which we collect for each step of model-generated samples. We compare these approaches in the context of a number of

*Equal contribution. Correspondence to nkushman@deepmind.com.

†Work performed at DeepMind, now at OpenAI.

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Workshop on MATH-AI.

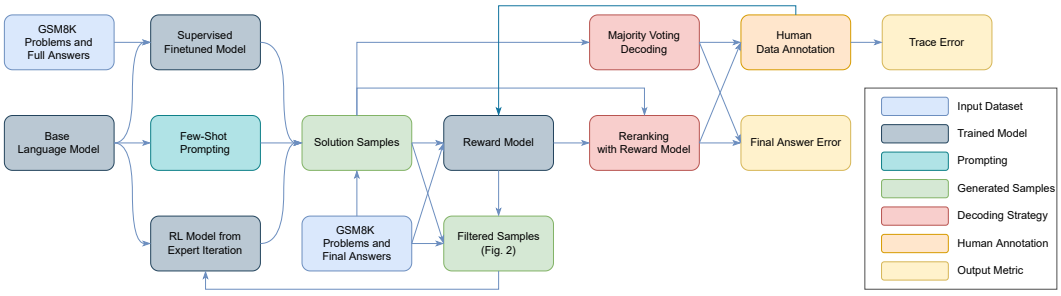

Figure 1: **Method Overview.** This schematic provides an overview of the various modeling and training components considered and how they fit together. Some details (covered in the text) are ommitted for readability.

different modeling and training components, including: few-shot prompting, supervised fine-tuning, reinforcement learning (RL) via expert iteration, and reward modeling for both reranking and RL.

Throughout, we consider two primary metrics: trace error rate, which measures how often the model makes any mistake in its reasoning trace according to human annotators, and final-answer error rate, which only considers the model's final answer and ignores the reasoning trace. By "reasoning trace" we refer to all steps of reasoning, including the last step which in GSM8K is the final numeric answer. While process-based approaches may provide multiple benefits, including encouraging human understanding of the problem domain, here we concentrate on investigating their effect on the trace error rate. We do so because trace error rate is directly measurable and of interest in many settings. For example, in educational settings, an answer without an (understandable) explanation may often confuse more than it explains. Recent findings suggest that outcome-based approaches often lack in this area. For example, work on natural-language-based reasoning (Zelikman et al., 2022; Creswell et al., 2022) suggests that models optimized exclusively for final-answer correctness can often produce the correct final answer, even when their generated reasoning traces are incorrect.

2 Methods

Our goal is to train a system for the sequence-to-sequence task (Sutskever et al., 2014) of taking the text of a problem as input and generating the text of an answer as output. For math word problems, the answer is a full reasoning trace: a newline-separated sequence of *steps*, where the last step is expected to provide the *final answer*. For GSM8K, the final answer is always an integer. Our approach broadly follows prior work on RL for language models (LMs) (Ziegler et al., 2019; Nakano et al., 2021; Menick et al., 2022). We use an LM as a *policy*, which maps the problem statement and steps-so-far to a next step. In the RL formalism, this treats each step as an action, and the observation is provided by all the tokens so far. The policy can be obtained through any of few-shot prompting, supervised finetuning (Section 2), or RL (Section 2). We also train LMs as *reward models* (Section 2), which score proposed full or partial completions from the policy, and can be used both for reranking samples from the policy, or as the source of rewards during reinforcement learning. All of our models are based on a 70 billion parameter pre-trained LM. (Hoffmann et al., 2022) In the following subsections, we describe how we train and assemble these components. See Fig. 1 for an overview and the appendix for additional details, discussion of the dataset, evaluation metrics and human data annotation procedures.

Supervised finetuning In supervised finetuning (SFT), we finetune an LM to maximize the log-likelihood of a sequence of target tokens, given a sequence of input tokens.

Reward models We evaluate two main approaches to training reward models (RMs) (Christiano et al., 2017; Ziegler et al., 2019; Menick et al., 2022), also known as verifiers (Cobbe et al., 2021). In both approaches, we implement the RM as an LM, trained to predict a binary label as either a 'correct' or 'incorrect' token after each step. In the *outcome-supervised RM* (ORM), the binary label for each step indicates whether the resulting final answer of that full sample matched the reference final answer, as proposed by Cobbe et al. (2021). For the *process-supervised RM* (PRM), the binary label

	Approach	Base model	Error rate (%) Trace	Final-answer
	Few-shot (Wang et al., 2022; Wei et al., 2022)	PaLM-540B	14.0	25.6
	Few-shot (Lewkowycz et al., 2022)	Minerva-540B	-	21.5
	Few-shot+Final-Answer RL (Zelikman, 2022)	GPT-J-6B	-	89.3
	Few-shot, ORM reranking (Li et al., 2022)	Codex-175B	-	16.8
	Zero-shot (Kojima et al., 2022)	InstructGPT-175B	-	59.3
	SFT, ORM reranking (Cobbe et al., 2021)	GPT-175B	-	45.0
No RM	Few-shot	Our Base-70B	-	41.5
No RM	Few-shot+Final-Answer RL	Our Base-70B	19.8 (7.9-31.7)	23.5
No RM	SFT+Final-Answer RL	Our Base-70B	12.1 (4.6-19.6)	20.2
No RM	SFT	Our Base-70B	11.4 (4.8-18.0)	22.3
RM Rerank	Few-shot, ORM reranking	Our Base-70B	-	27.8
RM Rerank	Few-shot+Final-Answer RL, ORM reranking	Our Base-70B	12.4 (2.1-22.8)	16.6
RM Rerank	SFT+Final-Answer RL, ORM reranking	Our Base-70B	3.7 (0.5-6.9)	14.2
RM Rerank	SFT, ORM reranking	Our Base-70B	4.4 (0.6-8.3)	14.8
RM Rerank	SFT, PRM reranking	Our Base-70B	3.5 (0.5-6.5)	14.1
RM-RL	Few-shot+ORM-RL, ORM reranking	Our Base-70B	5.5 (2.6-8.4)	13.8
RM-RL	SFT+ORM-RL, ORM reranking	Our Base-70B	3.4 (0.0-6.8)	12.7
RM-RL	SFT+PRM-RL, PRM reranking	Our Base-70B	3.8 (0.5-7.1)	12.9

Table 1: **Results overview.** We show trace and final-answer error rates. Trace error rates are averaged across raters. In parentheses, we provide a min-max range, depending on whether errors require both raters to agree (min) or just a single rater (max). While there is significant noise in the trace error rates, we can still observe general trends. Within each group, we list approaches in order from most outcome-based (top) to most process-based (bottom), other than the few-shot model, which we do not classify as there is no finetuning procedure to supervise.

after each step indicates whether the steps so far are correct. Because we lack reliable programmatic means for determining the correctness of intermediate steps, we use human annotations for these labels, as described in Appendix C.

For test-time decoding, we first sample $K = 96$ full solutions from the policy, and then select the best sample, either by ensembling or with an RM. When no RM is available, we use *majority voting* (Wang et al., 2022). For this, we first select the most common final answer from the K samples, then select a random sample from among those yielding this selected final answer. Otherwise, we use *RM-weighted* decoding, also called verifier-voting by Li et al. (2022). Here, we weight each sample according to the RM-estimated correctness probability, select the final answer with the largest total weight, and then select the sample with the highest RM score from those yielding the selected final answer.

RL via Expert Iteration All our RL experiments use expert iteration (Silver et al., 2017; Anthony et al., 2017) which alternates between (1) policy improvement, i.e. sampling from the current model and filtering these samples, and (2) distillation, training a new model using these filtered samples. The initial base policy can be either the SFT policy, or a 5-shot prompted version of our base LM.

We consider three versions of the policy improvement procedure. In the *Final-answer RL* approach, also called Self-taught Reasoner and proposed by Zelikman et al. (2022), we generate K full traces per problem and filter by final-answer correctness. In the *ORM-RL*/*PRM-RL* approaches we instead select the sample with the highest score according to the ORM/PRM model.

3 Results

Our results are summarized in Table 1. The ORM-RL and PRM-RL models achieve a final-answer error rate below 13%, improving on the 16.8% final-answer error for the current state-of-the-art model (Li et al., 2022). This is despite the fact that Li et al. (2022) use a base model which is better suited to math (Codex-175B), reporting 23.3% few-shot final-answer error, compared to 41.5% for

our few-shot base model. Our final-answer error rate is further reduced to 2.7% when the model is allowed to abstain on only 30% of questions. The ORM-RL and PRM-RL trace error rates, of 3.4% and 3.8% respectively, significantly improve on the 14% reported by the best prior work (Wang et al., 2022; Wei et al., 2022). Beyond these quantitative results, we highlight four key takeaways:

Supervising final-answer correctness alone suffices for low final-answer error rate. The SFT and Few-shot+Final-Answer RL models attain similar final-answer error rates both without an RM (22.3% vs. 23.5%) and with an ORM (14.8% vs. 16.6%). This is notable, as Few-shot+Final-Answer RL only requires supervision of the final answers, rather than the full reasoning traces.

ORM-supervised reward models approximate PRM labels. Despite the fact that ORMs are only trained to predict whether the final answer is correct, we find that ORM predictions tend to agree more with the PRM labels than with the ORM labels themselves (85% vs. 77% averaged over all steps). We suspect this is because it is simpler for the ORM to learn to recognize when steps are correct, than it is to check the answer by internally computing the final answer itself, however this effect may be specific to our domain.

Low trace error requires either process-based feedback or a reward model that emulates it. Table 1 shows that despite similar final-answer error rates, there is a significantly higher trace error rate for the outcome-based Few-shot+Final-Answer RL with ORM reranking vs. the process-based SFT with ORM/PRM reranking model (12.4% vs. 4.4%/3.5%). However, we find that when we train the few-shot RL model using an ORM (Few-shot+ORM-RL) rather than training directly against final-answer correctness, the trace error drops significantly from 12.4% to 5.5%, closing much of this gap. We believe this results from the previous finding, i.e. that the ORM is basically learning to emulate the PRM allowing the model to learn from emulated process-based feedback and resulting in relatively low trace error rates.

4 Related work

Math word problems have been a popular domain for studying reasoning in LMs (Kushman et al., 2014; Ling et al., 2017; Amini et al., 2019; Miao et al., 2020; Hendrycks et al., 2021; Cobbe et al., 2021; Ouyang et al., 2022; Kojima et al., 2022; Li et al., 2022; Brown et al., 2020; Chen et al., 2021). Several papers have demonstrated that few-shot prompting alone can lead to impressive performance on GSM8K (Chowdhery et al., 2022; Lewkowycz et al., 2022; Wei et al., 2022; Wang et al., 2022).

We focus on finetuning because we are interested in the effects of different feedback procedures, and because it significantly outperforms prompting alone for our base LM. The original GSM8K paper (Cobbe et al., 2021) demonstrated significant improvements from reward models or verifiers, and we use their ORM approach. Li et al. (2022) also study RMs and propose a heuristic-based step-aware RM, which slightly degrades performance on GSM8K, but boosts performance on a wide range of other benchmarks. We find that human evaluations of each step provide an improvement. We also use STaR (Zelikman et al., 2022) (referred to as Few-shot+Final-Answer RL throughout this paper), and show its GSM8K final-answer error can be reduced from their reported 89% to 23.5% through the use of a better base model (Hoffmann et al., 2022) and further reduced to 13.8% by using RM-based RL instead of their final answer RL procedure. In contrast to the above prior work, we not only show improved performance, but also provide a comprehensive comparison across different types of feedback, with a focus on trace error rate in addition to final-answer error rate.

Moving beyond math problems, much work studies multistep reasoning for LMs focusing either exclusively on outcome-based or process-based approaches (Lewkowycz et al., 2022; Ouyang et al., 2022; Perez et al., 2020; Shwartz et al., 2020; Wei et al., 2022; Kojima et al., 2022; Wu et al., 2021; Creswell et al., 2022; Nye et al., 2021; Zelikman et al., 2022). While most prior work on head-to-head comparisons of process- and outcome-based approaches has been on algorithmic tasks such as sorting numbers, which avoid working with human data (Graves et al., 2014; Reed and De Freitas, 2015; Li et al., 2016; Cai et al., 2017; Christiano et al., 2018). In contrast, we directly compare outcome-based and process-based techniques on a natural language task, and include a detailed analysis of trace error rates. WebGPT (Nakano et al., 2021) is the closest work to ours, however, we more comprehensively explore both process- and outcome-based supervision, additionally evaluating process-supervised RMs, the PRM-RL approach, and purely outcome-supervised RL policies (without SFT).

References

A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

A. Amini, S. Gabriel, P. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*, 2019.

D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

T. Anthony, Z. Tian, and D. Barber. Thinking fast and slow with deep learning and tree search. *Advances in Neural Information Processing Systems*, 30, 2017.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

J. Cai, R. Shin, and D. Song. Making neural programming architectures generalize via recursion. *arXiv preprint arXiv:1704.06611*, 2017.

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL `https://arxiv.org/abs/2107.03374`.

A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

P. Christiano, B. Shlegeris, and D. Amodei. Supervising strong learners by amplifying weak experts. *arXiv preprint arXiv:1810.08575*, 2018.

P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

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

A. Cotra. Without specific countermeasures, the easiest path to transformative ai likely leads to ai takeover, 2022. URL `https://www.alignmentforum.org/posts/pRkFkzwKZ2zfa3R6H/without-specific-countermeasures-the-easiest-path-to`.

A. Creswell, M. Shanahan, and I. Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

B. Dalvi, P. A. Jansen, O. Tafjord, Z. Xie, H. Smith, L. Pipatanangkura, and P. Clark. Explaining answers with entailment trees. *ArXiv*, abs/2104.08661, 2021.

M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and Ł. Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

D. Dohan, W. Xu, A. Lewkowycz, J. Austin, D. Bieber, R. G. Lopes, Y. Wu, H. Michalewski, R. A. Saurous, J. Sohl-dickstein, et al. Language model cascades. *arXiv preprint arXiv:2207.10342*, 2022.

R. El-Yaniv et al. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(5), 2010.

T. Everitt, V. Krakovna, L. Orseau, M. Hutter, and S. Legg. Reinforcement learning with a corrupted reward channel. *arXiv preprint arXiv:1705.08417*, 2017.

Y. Geifman and R. El-Yaniv. Selective classification for deep neural networks. *Advances in neural information processing systems*, 30, 2017.

M. Geva, D. Khashabi, E. Segal, T. Khot, D. Roth, and J. Berant. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics (TACL)*, 2021.

O. Goldman, V. Latcinnik, U. Naveh, A. Globerson, and J. Berant. Weakly-supervised semantic parsing with abstract examples. *arXiv preprint arXiv:1711.05240*, 2017.

N. Gontier, K. Sinha, S. Reddy, and C. Pal. Measuring systematic generalization in neural proof generation with transformers. *Advances in Neural Information Processing Systems*, 33:22231–22242, 2020.

A. Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

A. Guez, M. Mirza, K. Gregor, R. Kabra, S. Racanière, T. Weber, D. Raposo, A. Santoro, L. Orseau, T. Eccles, et al. An investigation of model-free planning. In *International Conference on Machine Learning*, pages 2464–2473. PMLR, 2019.

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

J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. v. d. Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

G. Irving, P. Christiano, and D. Amodei. Ai safety via debate. *arXiv preprint arXiv:1805.00899*, 2018.

Z. Kenton, T. Everitt, L. Weidinger, I. Gabriel, V. Mikulik, and G. Irving. Alignment of language agents. *arXiv preprint arXiv:2103.14659*, 2021.

T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

V. Krakovna, J. Uesato, V. Mikulik, M. Rahtz, T. Everitt, R. Kumar, Z. Kenton, J. Leike, and S. Legg. Specification gaming: the flip side of ai ingenuity, 2020. URL `https://deepmindsafetyresearch.medium.com/specification-gaming-the-flip-side-of-ai-ingenuity-c85bdb0deeb4`.

R. Kumar, J. Uesato, R. Ngo, T. Everitt, V. Krakovna, and S. Legg. REALab: An embedded perspective on tampering. *arXiv preprint arXiv:2011.08820*, 2020.

S. Kumar and W. Byrne. Minimum bayes-risk decoding for statistical machine translation. Technical report, JOHNS HOPKINS UNIV BALTIMORE MD CENTER FOR LANGUAGE AND SPEECH PROCESSING (CLSP), 2004.

N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, 2014.

A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, Y. Wu, B. Neyshabur, G. Gur-Ari, and V. Misra. Solving quantitative reasoning problems with language models, 2022. URL `https://arxiv.org/abs/2206.14858`.

C. Li, D. Tarlow, A. L. Gaunt, M. Brockschmidt, and N. Kushman. Neural program lattices. 2016.

Y. Li, Z. Lin, S. Zhang, Q. Fu, B. Chen, J.-G. Lou, and W. Chen. On the advance of making language models better reasoners. *arXiv preprint arXiv:2206.02336*, 2022.

W. Ling, D. Yogatama, C. Dyer, and P. Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*, 2017.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

J. Menick, M. Trebacz, V. Mikulik, J. Aslanides, F. Song, M. Chadwick, M. Glaese, S. Young, L. Campbell-Gillingham, G. Irving, et al. Teaching language models to support answers with verified quotes. *arXiv preprint arXiv:2203.11147*, 2022.

S.-y. Miao, C.-C. Liang, and K.-Y. Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, 2020.

A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862): 207–212, 2021.

R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

M. Nye, A. J. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan, A. Lewkowycz, M. Bosma, D. Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, 2022. URL `https://arxiv.org/abs/2203.02155`.

E. Perez, P. Lewis, W.-t. Yih, K. Cho, and D. Kiela. Unsupervised question decomposition for question answering. *arXiv preprint arXiv:2002.09758*, 2020.

E. Perez, D. Kiela, and K. Cho. True few-shot learning with language models. *Advances in Neural Information Processing Systems*, 34:11054–11070, 2021.

S. Polu and I. Sutskever. Generative language modeling for automated theorem proving, 2020. URL `https://arxiv.org/abs/2009.03393`.

M. Rauh, J. Mellor, J. Uesato, P.-S. Huang, J. Welbl, L. Weidinger, S. Dathathri, A. Glaese, G. Irving, I. Gabriel, et al. Characteristics of harmful text: Towards rigorous benchmarking of language models. *arXiv preprint arXiv:2206.08325*, 2022.

S. Reed and N. De Freitas. Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*, 2015.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

V. Shwartz, P. West, R. L. Bras, C. Bhagavatula, and Y. Choi. Unsupervised commonsense question answering with self-talk. *arXiv preprint arXiv:2004.05483*, 2020.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359, 2017.

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

A. Stuhlmüller and J. Byun. Supervise process, not outcomes. 2022. URL `https://ought.org/updates/2022-04-06-process`.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

O. Tafjord, B. D. Mishra, and P. Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *arXiv preprint arXiv:2012.13048*, 2020.

J. Uesato, R. Kumar, V. Krakovna, T. Everitt, R. Ngo, and S. Legg. Avoiding tampering incentives in deep RL via decoupled approval. *arXiv preprint arXiv:2011.08827*, 2020.

X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models, 2022.

J. Wu, L. Ouyang, D. M. Ziegler, N. Stiennon, R. Lowe, J. Leike, and P. Christiano. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*, 2021.

Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

E. Zelikman, Y. Wu, and N. D. Goodman. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022.

D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Dataset and evaluation metrics

We conduct all experiments on the GSM8K dataset (Cobbe et al., 2021), composed of grade school math word problems. We chose GSM8K because it is a competitive benchmark, and contains natural language reasoning traces. We focus on a single dataset, since the need to recruit human annotators with the domain expertise to accurately evaluate reasoning traces imposes a large up-front cost. We split out our own validation set of 256 examples from the original training set, which leaves us with 7118 training and 1319 test examples.

We report two main metrics for all methods evaluated on the GSM8K test set. *Final-answer error rate* is the fraction of problems for which the method does not produce the correct final answer. Because all final answers on GSM8K are integers, this can be measured with exact string matching. *Trace error rate* is the fraction of problems with correct final answers for which the method produces at least one incorrect reasoning step. We estimate this via human annotations of the correctness of each reasoning step, which is detailed in Appendix C.

We report final-answer and trace errors as two separate metrics because we are particularly interested in errors which remain undetected after checking easily verifiable metrics (in this case, final-answer errors). For example in an educational setting it is important to show a student the correct steps to get the answer, and we can easily filter out incorrect traces that lead to the wrong answer, but it is much more difficult to filter out incorrect traces that lead to the correct answer.

B Training Details

B.1 SFT

We finetune using AdamW (Loshchilov and Hutter, 2017) with a learning rate of 2×10^{-6} and a batch size of 256. We stop finetuning once the language modeling loss begins to increase on the validation set. For our SFT model, this happens after 70 steps, amounting to slightly more than 2 training set epochs.

B.2 Reward Models

Unless otherwise noted, for all approaches which include an ORM, we train the ORM using samples from the policy for that approach, taking $K = 96$ samples with temperature 1.0,

We follow Cobbe et al. (2021) and regularize with dropout, with a dropout parameter of 0.1, and otherwise reuse the hyperparameters used for SFT from Section 2. To speed up learning in the SFT-based approaches, we initialize the ORM training using the SFT model parameters, while for the few-shot based approaches we initialize from the base pretrained LM. For the PRM, we annotate 3 samples per problem from the SFT policy, restricting to problems where the SFT majority prediction (see **??**) was incorrect, in order to make the most of our human annotation budget. Due to the small size of our human-annotated dataset (1560 full solutions), we initialize the PRM parameters to the ORM parameters and lower the learning rate to 1×10^{-7}. The RM loss curves have some fluctuation, and so we select the RM with the best validation loss before 2000 steps.

B.3 Decoding

We sample with temperature $T = 1.0$, and use the syntax from Cobbe et al. (2021) to allow the model to decide when to use a calculator. In early experiments, we also tried RM reranking after each generated step (rather than the full solution), but found that this led to slightly worse performance, increasing final-answer error by 1-2%.

Formally, in RM-weighted decoding we select the final answer $f^* = \arg\max_f \sum_{y_i:\text{final\_ans}(y_i)=f} \text{rm\_prob}(y_i)$, where $y_1, \ldots, y_K$ are the model samples, then select the best sample according to $y^* = \arg\max_{y:\text{final\_ans}(y)=f^*} \text{rm\_prob}(y_i)$. This works slightly better compared to simply selecting the sample with the highest RM score (improving final-answer error rate about 1% with the SFT model, and slightly more with the RL models). However, we note that both majority voting and RM-weighted decoding are slightly less general due to their reliance on exact string-matching between final answers.

B.4 RL via Expert Iteration

We note that, aside from the 5 random training examples used for the prompt, none of the few-shot-based approaches ever use the intermediate reasoning steps provided in the GSM8K dataset, our human annotations, or any models derived from this data. When initializing from the SFT model, we follow Polu and Sutskever (2020) and reuse expert samples from each iteration, so that our training set grows each epoch. We do not do this with few-shot approaches because in that setting, the samples from the early epochs have many trace errors which we do not want the RL model to imitate. Correspondingly, there are several minor implementation differences between the two cases, which we note below

Policy Improvement For the few-shot version, we select all traces yielding the correct final answer, while for the SFT-based version, we only use one randomly chosen sample per problem. In the *ORM-RL* approach, we generate K full traces per problem, and select the sample with the highest score according to the ORM model. In the *PRM-RL* approach, we instead treat each step as an individual episode. At each step, we generate K candidate steps, select the candidate with the highest PRM score, and continue from the selected step until the model outputs a step with the final answer indicator text, or a maximum of 15 steps. We set $K = 96$ across all experiments. For few-shot-based approaches, we retrain the RM after every expert iteration. For SFT-based approaches, we skip this step and use a fixed RM, since somewhat surprisingly, this did not make a significant difference in preliminary experiments.

Distillation For distillation, we use the same hyperparameters as SFT. As with SFT, we apply early stopping by validation loss, where our validation set is constructed from expert policy samples on the validation set. For SFT-based approaches, we initialize with the SFT parameters at each distillation step, while for few-shot-based approaches, we initialize with the base model parameters.

C Data annotation

As discussed in Section 2, the PRM is trained on stepwise labels indicating whether the steps so far are correct. To collect this data, we present human annotators with the problem statement, the reference solution from GSM8K, and the generated model solution, and ask them to indicate the first model step with a major mistake, if any exist. Our instructions define a major mistake as "a step where the information expressed is incorrect, or it would no longer be possible to reach the correct solution without undoing that step". From these annotations, we can label every step with a binary label indicating whether the steps so far are correct: all steps before the first major mistake are labeled 'correct', while the remainder are labeled 'incorrect'.

We applied a small amount of dataset cleaning by removing samples from annotators with low inter-annotator agreement (measured on the 20% of solutions where we used duplicate labelling), as well as those from GSM8K problems flagged by annotators as ambiguous. This removed about 20% of our data, leaving annotations for 1560 model samples across 530 training set problems, corresponding to 9856 step-level binary labels. For the validation set, we used the same procedure, but added duplicate labelling and a manual pass by the paper authors to resolve inter-annotator disagreements. Our validation set contained 162 model samples, with 913 total steps. For evaluation, we used 200 problems with correct final answers per model. This was done for each of the 10 models in Table 1, again with duplicate labelling.

Participants and pay The full details of our study design, including compensation rates, were reviewed and approved by our independent ethical review committee. All participants provided informed consent prior to completing tasks and were reimbursed for their time. It is our policy that researchers must pay workers/participants at least the living wage for their location.

Training dataset problems For constructing the PRM training dataset, we used samples from the SFT model. Due to a limited annotation budget, we only annotate problems where the SFT majority voting prediction is incorrect, since this focuses training on difficult problems, and still includes a mix of correct and incorrect samples due to annotating 3 model solutions per problem.

Quality assurance Since evaluating the accuracy of solutions to mathematical problems is a special skill, we ran a preliminary qualification study before using annotations from participants for training or evaluation. For the qualification annotation tasks, we selected model solutions where three authors unanimously agreed on the first major mistake, and required participants to annotate at least 3 out of 4 such solutions correctly. In total, 21 / 91 candidates were included in our annotator pool.

Additionally, we used duplicate annotations for 20% of the training problems. We removed ratings from annotators who had an inter-annotator agreement rate below 75% on doubly-rated problems. On manual inspection, we found that these annotators had typically made errors in this disagreeing cases. This removed data from 4 / 21 annotators, amounting to 21% of our originally labelled training data, leaving us with 530 annotated problems (compared to 675 originally).

After quality assurance steps, using the same set of duplicated problems, we measure inter-rater agreement rate of 92% and Cohen's κ of .915 for the task of predicting the first incorrect step. Note that this estimate will be slightly biased upwards as we filtered raters based on this same set, but as the inter-rater agreement rate is fairly bimodal across raters, this effect should be relatively small.

For the evaluation, we gather duplicate annotations for all ratings, and see an inter-rater agreement of 87%, with Cohen's κ of .34 on the binary task of labelling the full trace as correct. Inter-rater agreement is significantly lower than on the training set. We attribute this to the fact that for evaluation we only annotate traces with the correct final answer, and observe that when the final answer is correct, mistakes in the reasoning can be quite subtle, while when the final answer is incorrect the reasoning mistakes are often more clear cut.