A Study on Improving Reasoning in Language Models

Yuqing Du*
UC Berkeley

Alexander Havrilla
Georgia Tech

Sainbayar Sukhbaatar
Meta AI Research

Pieter Abbeel
UC Berkeley

Roberta Raileanu
Meta AI Research

Abstract

Accurately carrying out complex reasoning is a crucial component of deployable and reliable language models. While current language models can exhibit this capability with few-shot guidance, accurate reasoning is primarily restricted to larger model sizes. In this work, we explore methods for improving the reasoning capabilities of smaller language models which are more deployable than their larger counterparts. Specifically, we look at variations of supervised learning, online reinforcement learning with PPO, and distillation from larger models. Surprisingly, for reasoning tasks such as CommonsenseQA [26] and GSM8K [2] we find that simple filtered supervised learning often outperforms reward-conditioned supervised learning, and that simple iterative supervised learning performs on par with online reinforcement learning.

1 Introduction

Reasoning is a crucial aspect of human decision making and intelligence. Recent work has shown that it is possible to elicit coherent reasoning from language models, especially when the models make use of step-by-step reasoning, or 'chains of thought' [23, 32, 17]. Despite these models not being explicitly trained to learn reasoning skills, the combination of a vast pretraining corpus and sufficiently large model size has led to surprisingly performant models on various reasoning tasks [31]. This is a particular capability of interest as reasoning, problem solving, and decision making capabilities can be seen as hallmarks of human intelligence [24].

However, impressive reasoning capabilities are currently primarily restricted to *large* language models. For example, [32] find that chain-of-thought prompting is an emergent property, only yielding "performance gains when used with models of $\sim$100B parameters". In this work, we are interested in studying methods for improving reasoning in smaller language models, *i.e.* fewer than 10B parameters. Teaching smaller models to reason well can enable faster deployment, removing the burden of serving large language models (LLMs) in practice. For example, a mobile phone or household robot is likely unable to host the hardware necessary to serve the most capable LLMs, yet we would like them to have the same general reasoning abilities as current large models.

As reasoning tasks can often have a correct outcome (*e.g.* there exists a correct final answer for a math problem), we assume we are operating in the setting where we have access to outcome-based supervision. That is, we assume access to ground truth final answers for reasoning tasks. Given this assumption, how can we best train smaller language models to improve their reasoning abilities on unseen questions? We explore two data sources for teaching smaller models to reason: distillation from a more capable teacher model, and self-taught reasoning. We also explore two methods of self-taught reasoning: iteratively applying supervised learning from an outcome-filtered dataset

*Correspondence to yuqing_du@berkeley.edu

I Can't Believe It's Not Better Workshop at NeurIPS 2023.

and online reinforcement learning via policy gradient method. Concretely, we conduct a study on improving the reasoning capabilities of smaller (6-7B parameter) language models, comparing the effects of data sources and learning schemes. We focus on two tasks: commonsense reasoning via CSQA [26] and mathematical reasoning via GSM8K [2]. We find:

- While prior work has found reward-conditioned supervised learning to be effective in other settings [8, 7], surprisingly, we find that for reasoning, conditional supervised finetuning is often more harmful than helpful.
- Iterative policy improvement via expert iteration or policy gradient have a similar test performance, resulting in comparable gains on reasoning tasks relative to supervised learning.
- If larger models are accessible, one should leverage distillation of reasoning traces generated by the larger (teacher) model, which can be combined with policy improvement methods.

2 Related Works

Reasoning in Language Models Many benchmark tasks have been proposed for assessing language model reasoning capabilities, such as: commonsense reasoning [26], mathematical reasoning [2, 19, 3], multi-hop reasoning [34, 28], or reasoning about external knowledge sources [27]. Reasoning can be elicited from sufficiently large language models with chain-of-thought prompting [32], which provides the language model few-shot examples of correct reasoning steps and a correct final outcome.

Improving Reasoning in LMs There exists a large body of work studying methods for improving complex reasoning in language models. In cases where external sources of correct reasoning traces are available, one can finetune on a dataset of questions, rationales, and answers. These rationales can be human-written [2], synthetically generated [17, 9], or generated by larger, more capable models [5, 30, 4, 13, 10, 14]. In cases where access to expert data is limited, a complementary method is to provide sparser reward supervison. Reward functions can be used to evaluate final outcomes only, or a combination of intermediate reasoning steps and final outcomes [29, 11]. Such reward functions can be used for filtered or ranked supervised learning [35, 2, 29], reinforcement learning [36, 11, 33], or a combination of the two [15, 18]. These prior works have shown the benefit of using reinforcement learning on top of a supervised finetuned model, especially with respect to alignment with human feedback. In contrast, we focus on directly comparing policy improvement on reasoning tasks with iterative outcome supervision either via expert iteration [29] or policy gradient methods [21].

3 Improving Reasoning in LMs

Suppose we are given a dataset $\mathcal{D}$ of problems with x and corresponding answers y – i.e. $\mathcal{D} = \{(x_i; y_i)\}_{i=1}^{N}$. Let f be the model we are training (i.e. the smaller student model). Let $\hat{z}_i$ be the model-generated reasoning trace (e.g. CoT) for question x_i, leading to model generated answer $\hat{y}_i$. In this section, we formalize the different finetuning schemes. For details, see Appendix A.

(1) Fewshot prompting. Do no training and condition the model f on a few examples of $(x_i; z_i; y_i)$ where z_i is a correct reasoning trace. Typically a small number of samples, *e.g.* $i < 10$.

(2) Supervised finetuning on labels (SFT-L). Simply finetune on the paired question and answers directly. We apply the standard per-token negative log likelihood loss.

(3) Supervised finetuning on *external* reasoning traces and labels (Distillation). Simply finetune on the combined question, reasoning trace, and answer. We can apply the same standard negative log likelihood loss. For distillation, these reasoning traces come from a teacher model.

(4) Supervised finetuning on positive self-reasoning traces and labels (SFT-filter). Since we only have outcome labels y_i, we can few-shot prompt f to generate its own reasoning traces and corresponding final answers, and filter for reasoning traces where the final answer matches the ground truth label (*i.e.* positive samples). This can be applied iteratively for continual self-improvement.

(5) Conditional supervised finetuning on positive and negative self-reasoning traces and labels (CSFT). In the previous loss, we only utilized positive samples and discarded negative ones. However, ideally our model should be able to learn from negative samples as well. One simple way of learning different distributions from positive and negative samples is through reward-conditioning. We label reward using special tokens `<|good|>` and `<|bad|>`, as in prior work [8]. Following the procedure

	GPT-J (6B)		Llama-2 (7B)	
	Test Accuracy	Data Used	Test Accuracy	Data Used
Fewshot prompting	35%	-	63%	-
SFT- lter	47%	33%	70%	62%
CSFT	38%	100%	66%	100%
SFT-L	52%	100%	75%	100%
Distillation	60%	76%	74%	76%

(a) CSQA Single-Iteration evaluation.

	GPT-J (6B)		Llama-2 (7B)	
	Test Accuracy	Data Used	Test Accuracy	Data Used
Fewshot prompting	3.0%	-	14%	-
SFT- lter	3.1%	2.3%	17%	15%
CSFT	2.8%	100%	14%	100%
SFT-L	4.7%	100%	8.2%	100%
Distillation	10.6%	58%	31%	58%

(b) GSM8K Single-Iteration evaluation.

Table 1: Test Accuracy shows accuracy on each respective test set, using greedy decoding with 1 attempt per question. Data Used shows the percentage of the training set questions used for netuning. We bold the best self-improved results (not including Distillation since it relies on a separate model). For CSQA, where intermediate reasoning steps may not be crucial for getting the correct answer, netuning on the full dataset of labels is preferable. On the other hand, for GSM8K, where generating reasoning is necessary for arriving at the correct answer, SFT- lter is preferable.

from (SFT- lter) , we initially prompt the language model to generate reasoning traces and answers for a set of questions. We prepend each positive and negative example with <|good|> or <|bad|> respectively and at test time, we condition the model on <|good|> only.

(6) Policy Gradient. We consider the sparse reward case where we apply the reward at the nal token, $r_T = 1(\hat{y}_i = y_i)$. We can apply standard deep RL algorithms to this setting; we use PPO [21].

For all of the above settings, we train on the combined question, reasoning trace, and answer. One could also mask out the question since the model does not need to learn to generate questions.

4 Experiments

We consider Single-Iteration Policy Improvement (Section 4.1) and Multi-Iteration Policy Improvement (Section 4.2). We use `iteration' to refer to the number of times we generate new data with the updated model for netuning. The former focuses on the case where we only do one pass of data generation, while the latter contains cases where we can continually generate new data for training (i.e., iteratively applying SFT- lter/CSFT or online RL). See Appendix B for training details.

4.1 Single-Iteration Policy Improvement

First, we consider the case where we can only netune the model with a single training iteration, not generating new training data as the model is updated. In this section, we try to answer the following question: what data source will result in the highest test-time performance if we can only netune the language model once? We evaluate four methods for netuning and use a few-shot prompting baseline: 1) one iteration of ltered supervised netuning (SFT- lter), 2) one iteration of conditional netuning (CSFT), 3) one iteration of netuning on questions and labels only (SFT-L), and 4) distilling reasoning traces from the more capable Llama-2 65B model, after ltering for correctness. See Section 3 for details on the instantiation of each method.

Table 1 shows the test-time accuracies across all methods. Unsurprisingly, distillation is generally the best approach if one has access to a better model. If not, we nd that the alternative approaches vary in effectiveness depending on the task. For CSQA, where intermediate reasoning steps may not be crucial for getting the correct answer, netuning on the full dataset of labels is preferable.

	Test Acc. (Whole Dataset)	Test Acc. (Self-Generated)
No errors (SFT-Filter)	33.0%	17%
No errors (CSFT)	32.4% (-0.6%)	17% (+0%)
1 step error (CSFT)	35.3% (+2.3%)	20% (+3%)
2 steps error (CSFT)	35.4% (+2.4%)	18% (+0.9%)

Table 2: Llama-2 (7B) GSM8K test set accuracy, (CSFT using synthetic negatives. Note that 1 and 2-step error conditions doubles the amount of training data (each question now has both a positive and negative reasoning trace). The left column uses the dataset-provided reasoning traces (7470), while the right column only uses the filtered positive reasoning traces (1102/7470).

On the other hand, for GSM8K, where generating reasoning is necessary for arriving at the correct numerical answer, we find (SFT-filter) is preferable. Interestingly, we find that conditional finetuning with additional negative samples generally hurts reasoning performance compared to simple positive filtering. Even if we balance the number of positive and negative examples (Table 6), we find that evaluation performance either remains the same or decreases.

When is conditional training helpful for reasoning? One question we would like to better understand is why conditioning on negative reasoning traces harms performance, even though this method has generally been found to be helpful for alignment and dialogue [8, 12]. Is it the case that there are too many diverse ways of reasoning poorly, making learning a conditional distribution difficult from good and bad self-generated reasoning traces difficult? Since we have ground truth, human-written reasoning traces for GSM8K, we investigate providing augmented synthetic negative samples, that may allow us to better understand what `useful' negative samples look like. Here, we hypothesize that negative samples that are smaller deviations from positives ones may be easier to learn. We construct negative samples as follows: 1-step—only the final outcome is incorrect, but all reasoning before is sound, or 2-step—the final outcome is incorrect and the final step is missing, but all preceding reasoning is sound. See Table 7 for examples. With this approach, (CSFT) is beneficial (see Table 2). This suggests that more closely aligned negatives can be beneficial. However, constructing such examples may be difficult if we do not have access to human written or human annotated steps for reasoning. One potential avenue for interesting further research is to use the language model itself to edit similar small synethetic errors on correct self-generated traces. Generally, converting correct reasoning traces into incorrect ones is easier than the other way around so we could expect language models to be useful at generating such data.

Single-iteration Takeaways:

1. CSFT with incorrect generations has a neutral effect at best. Training with synthetic negatives to boost the amount of data is potentially helpful.
2. In cases where complex reasoning is needed to get to the correct final answer (e.g. GSM8K), it is better to train on a smaller filtered dataset of reasoning traces with the final answers (SFT-filter) rather than the whole dataset of final answers directly (SFT-L). One potential alternative we leave for future work is a combination of the two: combining filtered reasoning trace and answers with answers-only for the questions that the model was unable to answer.
3. In cases where a larger model is available, distillation of filtered correct answers and reasoning traces is often better than training on the whole dataset of answers only (SFT-L), even though the latter uses the whole training set of questions.

4.2 Multi-Iteration Policy Improvement

Next, we consider the case where we can finetune the model multiple times. In this section, we try to answer the following question: is it preferable to iteratively repeat (SFT-filter) or use PPO?

Iterative Supervised Finetuning. Building on the above results, we can also iteratively apply the above techniques to continually improve reasoning capabilities. As in STaR [34], we reset the model to the base model at each iteration and do not accumulate data over iterations. In Figure 1, we

(a) Y-axis shows test set accuracy on CSQA. (b) Y-axis shows test set accuracy on GSM8K.

Figure 1: Finetuning Llama2 (7B) with using SFT- lter , comparing different scheduling schemes. Following STaR's xed (40 steps to start, increase by 20% each iteration) scheduling proposed in the paper leads to better test performance at convergence than using early stopping each iteration (STaR-ES). Note that (+rationalization) means we apply the rationalization method from STaR, where we generate reasoning in hindsight after being given the correct ground truth answer.

(a) Y-axis shows test set accuracy on CSQA. (b) Y-axis shows test set accuracy on GSM8K.

Figure 2: Finetuning Llama-2 (7B) with PPO. We compare three different model initializations (base, SFT- lter , and distilled), and various augmentations of the base algorithm. Of the changes tried, removing GAE and detaching the value head were most helpful for improving training stability.

compare with an early stopping scheduler based on a held out validation set loss, rather than the xed schedule proposed in the original work. Interestingly, we nd that early stopping leads the model to generally converge to a lower test accuracy, suggesting that careful selection of a validation set is necessary or use of early stopping may be unsuitable in combination with STaR. This suggests that iterative netuning can be sensitive to the prescribed scheduling and data being trained on. This can possibly be related to the general problem of a model collapse, with generations degrading in quality as it continues to train on synthetic samples [22]. For additional experiments on applying CSFT iteratively, see Appendix D.1.

Policy Gradient. We explore a range of different methods to see if they improve policy learning. These are primarily based on the hypothesis that improving reasoning with RL is challenging due to the dif culties of value learning in the language modelling setting.

- Removing GAE [20] by setting = 1: avoid using predicted values for advantage estimation if our value function is inaccurate.
- Detaching value head gradients: prevent value estimation errors from affecting underlying language modelling abilities by not backpropagating value updates through the language model.
- Separate value heads: train separate value heads for the task reward and KL penalty, with the intuition that that is easier than learning one shared value function.
- Resetting model: to prevent the model from reaching a local minima, we can occasionally reset model parameters [15]. We only reset the nal layer of the language model (before the value head) and update 10x more times on the same batch of data whenever a reset occurs.

In the PPO setting, we compare three different model initializations: (1) the base Llama-2 (7B) model, (2) a Llama-2 (7B) model that has gone through one iteration of SFT- lter , and (2) a Llama-2 (7B) model that has gone through one iteration of ltered distillation from Llama-2 65B.

Two big challenges we found while using PPO were reward collapse—where the model performance would suddenly drop to 0, and lack of generalization—where performance would increase on the training set but remain constant for the test set. While decreasing learning rate and increasing the KL penalty helped with the collapse issue, they did not fix the problem entirely. Instead we found that the change that made the biggest difference is using LoRA [6], corroborating results from [25]. Of the additional augmentations we tried above, we find that detaching the value head gradients and removing GAE were most helpful for slightly stabilizing training. This motivates further investigations into whether it would be more effective to have a separate value model entirely (if one has sufficient memory space), or even to pretrain the value head directly. Removing GAE also lead to similar improvements in performance, albeit being less sample efficient. The other augmentations generally hurt performance.

Multi-Iteration Takeaways:

1. STaR is sensitive to correct scheduling – model can converge to lower performance at test time. Rationalization is not helpful if the reasoning generated in hindsight is incorrect.
2. LoRA [6] was necessary for seeing meaningful generalization (compared to fully finetuning between 1-16 layers).
3. Detaching the value head and removing GAE helped with training stability.
4. We find that we are able to improve reasoning performance on the test set substantially from the base model, without having to initialize with SFT-Filter beforehand. While this does not outperform the models that are initialized with SFT-Filter, it shows that PPO can lead to the model to learn to reason correctly at test time about questions that the base model initially could not.

4.3 Comparing SFT-Filter and RL

	CSQA	GSM8K
STaR	76.2%	29.0%
STaR (+rationalization)	77.4%	14.1%
PPO	77.0%	30.0%

Table 3: Best Test Accuracy, Finetuning Llama-2 (7B) with iterative training methods. Ultimately, the best performing setting with either multi-iteration method are similar.

Taking the best checkpoints of either method, both multi-iteration SFT-Filter and PPO perform similarly at test time. RL does not require the STaR-specific scheduling, but is much more prone to mode collapse and may result in less diverse outputs. The RL setting does more exploration (through higher temperature sampling at training time and retaining negative samples), while SFT-Filter only uses greedy sampling and throws away negative samples.

5 Conclusion

We investigate various methods for improving the reasoning capabilities of finetuned language models. For a single training iteration, we compare distillation from a more capable model, conditional supervised learning, and filtered supervised learning. This work raises many questions for further work. In the SFT-Filter case, it would be meaningful to explore how sensitive the method is to the STaR scheduling. Furthermore, getting more diverse samples with higher temperatures and prompting with different schemes could also improve the performance of applying multiple rounds of SFT-Filter. Further investigations into which components of PPO are necessary for improving at reasoning tasks is also an interesting avenue for further work. Many implementation details in PPO were included for performance in control environments, but may not be necessary in the language modelling case (where our base model is already somewhat capable, we do not use discounting, etc.).

References

[1] L. Castricato, A. Havrilla, S. Matiana, D. V. Phung, A. Tiwari, J. Tow, and M. Zhuravinsky. trlX: A scalable framework for RLHF, June 2023.

[2] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training veri ers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

[3] 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.

[4] N. Ho, L. Schmid, and S.-Y. Yun. Large language models are reasoning teachers. arXiv preprint arXiv:2212.10071, 2022.

[5] C.-Y. Hsieh, C.-L. Li, C.-K. Yeh, H. Nakhost, Y. Fujii, A. Ratner, R. Krishna, C.-Y. Lee, and T. P ster. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. arXiv preprint arXiv:2305.02301, 2023.

[6] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685, 2021.

[7] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher. Ctrl: A conditional transformer language model for controllable generation. arXiv preprint arXiv:1909.05858, 2019.

[8] T. Korbak, K. Shi, A. Chen, R. Bhalerao, C. L. Buckley, J. Phang, S. Bowman, and E. Perez. Pretraining language models with human preferences. ArXiv, abs/2302.08582, 2023.

[9] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. 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. ArXiv, abs/2206.14858, 2022.

[10] L. H. Li, J. Hessel, Y. Yu, X. Ren, K.-W. Chang, and Y. Choi. Symbolic chain-of-thought distillation: Small models can also" think" step-by-step. arXiv preprint arXiv:2306.14050, 2023.

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

[12] H. Liu, C. Sferrazza, and P. Abbeel. Chain of hindsight aligns language models with feedback. ArXiv, abs/2302.02676, 2023.

[13] L. C. Magister, J. Mallinson, J. Adamek, E. Malmi, and A. Severyn. Teaching small language models to reason. arXiv preprint arXiv:2212.08410, 2022.

[14] S. Mukherjee, A. Mitra, G. Jawahar, S. Agarwal, H. Palangi, and A. H. Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. ArXiv, abs/2306.02707, 2023.

[15] 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.

[16] E. Nikishin, M. Schwarzer, P. D'Oro, P.-L. Bacon, and A. Courville. The primacy bias in deep reinforcement learning. In International conference on machine learning, pages 16828–16847. PMLR, 2022.

[17] 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.

[18] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems, 35:27730–27744, 2022.

[19] A. Patel, S. Bhattamishra, and N. Goyal. Are nlp models really able to solve simple math word problems? arXiv preprint arXiv:2103.07191, 2021.

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

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

[22] I. Shumailov, Z. Shumaylov, Y. Zhao, Y. Gal, N. Papernot, and R. Anderson. Model dementia: Generated data makes models forget. arXiv preprint arXiv:2305.17493, 2023.

[23] 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.

[24] R. J. Sternberg and S. B. Kaufman. The Cambridge handbook of intelligence. Cambridge University Press, 2011.

[25] S. Sun, D. Gupta, and M. Iyyer. Exploring the impact of low-rank adaptation on the performance, ef ciency, and regularization of rlhf. arXiv preprint arXiv:2309.09055, 2023.

[26] A. Talmor, J. Herzig, N. Lourie, and J. Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. arXiv preprint arXiv:1811.00937, 2018.

[27] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal. Fever: a large-scale dataset for fact extraction and veri cation. arXiv preprint arXiv:1803.05355, 2018.

[28] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal. Musique: Multi-hop questions via single-hop question composition. Transactions of the Association for Computational Linguistics, 10:539–554, 2022.

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

[30] P. Wang, A. Chan, F. Ilievski, M. Chen, and X. Ren. Pinto: Faithful language reasoning using prompt-generated rationales. arXiv preprint arXiv:2211.01562, 2022.

[31] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al. Emergent abilities of large language models. arXiv preprint arXiv:2206.07682, 2022.

[32] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35:24824–24837, 2022.

[33] 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.

[34] 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.

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

[36] 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 Details on Schemes for Improving Reasoning

Suppose we are given a dataset D of problems with x and corresponding answers y – i.e. $D = \{(x_i, y_i)\}$. Let f be the model we are training (i.e. the smaller student model). Let z_i be the model-generated reasoning trace (e.g. CoT) for question x_i, leading to model generated answer $\hat{y}_i$. In this section, we formalize the training objectives for different finetuning schemes.

(1) Fewshot prompting. We do no training, but condition the model on a few examples of (x_i, z_i, y_i) where z_i is a correct reasoning trace. Typically a small number of samples is < 10.

(2) Supervised finetuning on labels (SFT-L). We do not use reasoning traces, instead simply finetune on the paired question and answers directly. This is possible given our assumption of access to ground truth outcome labels. Let $w_i = (x_i, y_i)$ be the concatenation of each question, answer pair. We apply the standard per-token negative log likelihood loss as follows

$$L_{\text{label}} = \frac{1}{|D|} \sum_{i \in D} \sum_{t} \log f_\theta(w_{i,t} | w_{i,1:t-1})$$

(3) Supervised finetuning on external reasoning traces and labels (Distillation). In cases where we have access to reasoning traces z_i, let $w_i = (x_i, z_i, y_i)$. We can apply the same standard negative log likelihood loss as in (2). For distillation, these reasoning traces come from a teacher model.

(4) Supervised finetuning on positive self-reasoning traces and labels (SFT-filter). Since we only have outcome labels y_i, we can few-shot prompt f to generate its own reasoning traces and corresponding final answers, and filter for reasoning traces where the final answer matches the ground truth label (i.e. positive samples). In this case, let $w_i = (x_i, \hat{z}_i, \hat{y}_i)$, where $\hat{z}_i, \hat{y}_i \sim f(\cdot|x_i)$. This can be applied iteratively for continual self-improved reasoning.

$$L_{\text{SFT-filter}} = \frac{1}{|D|} \sum_{i \in D} \sum_{t} \mathbb{1}(\hat{y}_i = y_i) \log f_\theta(w_{i,t} | w_{i,1:t-1})$$

As noted in [35], this objective is analogous to vanilla policy gradient with a sparse, binary reward.

(5) Conditional supervised finetuning on positive and negative self-reasoning traces and labels (CSFT). In the previous loss, we only utilized positive samples and discarded negative ones. However, ideally our model should be able to learn from negative samples as well, for example, by decreasing the likelihood of generating invalid reasoning traces. One simple way of learning different distributions from positive and negative samples is through reward-conditioned finetuning. We label for reward by using special tokens <|good|> and <|bad|>, as in prior work [8].

We follow a similar procedure as (SFT-filter) where we initially prompt the language model to generate reasoning traces and answers for a set of questions. However, in this case we do not discard incorrect samples that led to the wrong outcome. We prepend each positive and negative example with <|good|> or <|bad|> respectively, encouraging the model to learn a conditional distribution over positive and negative samples. At test time, we condition the model on <|good|> only. In this case, let $w_i = (x_i, c_i, \hat{z}_i, \hat{y}_i)$, where $\hat{z}_i, \hat{y}_i \sim f(\cdot|x_i)$ and c_i is <|good|> if $\hat{y}_i = y_i$ and <|bad|> otherwise. As above, this process can be applied iteratively for continual self-improved reasoning.

(6) Policy Gradient. Given that we have access to the ground truth label, we can also use reinforcement learning for teaching LMs to reason. Let r_t be the reward at t-th token. For the sparse reward case, we apply the reward at the final token. That is, the reward is all zeros except at $t = T$, where $r_T = \mathbb{1}(\hat{y}_i = y_i)$. We can apply standard deep RL algorithms to this setting; we use PPO [21]. To reduce variance, we learn a value function v and estimate the advantage $\hat{A}_t = \sum_{t' > t} r_{t'} - v(w_{1:t})$

$$L_{\text{PPO}} = \frac{1}{|D|} \sum_{i \in D} \sum_{t} \log f_\theta(w_{i,t} | w_{i,1:t-1}) \hat{A}_t$$

In addition to the task outcome-based reward, we can also add a per-token KL penalty to the reward to prevent the finetuned model from deviating too far from the pretrained model [36].

Note that for all of the above settings, we train on w_i as the whole combined question, reasoning trace, and answer. An alternative approach would be to only compute loss on the generated tokens, since the model does not need to learn to generate questions.

B Training Details

Supervised learning: To generate the full training set we need to sample reasoning traces and answers for all questions in the dataset. To do so, we few-shot prompt the model with 6-7 example questions, reasoning traces, and answers, and sample greedily to generate new reasoning traces and answers. For STaR-filter we filter for correct answers only, and for STaR-SFT we annotate each reasoning traces with whether it was <|good|> or <|bad|>. We use the same few-shot prompting scheme during evaluation as well. To determine which checkpoint to evaluate, we use early stopping based on loss on a held-out validation set (taken from the generated training set above) unless otherwise stated (e.g. training with STaR for fixed total step sizes in Section 4.2). Specifically, for each training dataset generated, we hold out 10% of the samples for validation. We use the HuggingFace trainer for finetuning and we do not append few-shot samples to each example in the training set.

Reinforcement Learning (PPO): We use TRLX [1] for PPO training. In addition to the default settings, we use best-of-4 sampling (i.e. we generate 4 rollouts per question and only the highest reward sample), with temperature 0.7 during generation to encourage exploration. At evaluation time, we do the same greedy sampling as in the SFT case. Due to memory limits, we only use 1-shot examples for RL.

B.1 Prompts

We use the same few-shot prompts as in STaR [38]. Due to memory issues with PPO training, we only use 1-shot prompting. For all other experiments, we use few-shot prompting.

CSQA prompt:

Q: What do people use to absorb extra ink from a fountain pen? Answer Choices: (A) shirt pocket (B) calligrapher's hand (C) inkwell (D) desk drawer (E) blotter A: The answer must be used to absorb extra ink. Blotters are designed to absorb liquids. Therefore, the answer is blotter (E). Finish[E]

Q: What home entertainment equipment requires cable? Answer Choices: (A) radio shack (B) substation (C) television (D) cabinet (E) desk A: The answer must require cable. Cable is used to provide satellite channels to televisions. Therefore, the answer is television (C). Finish[C]

Q: The fox walked from the city into the forest, what was it looking for? Answer Choices: (A) pretty flowers (B) hen house (C) natural habitat (D) storybook (E) dense forest A: The answer must be a reason for a fox to go into the forest. The forest is a fox's natural habitat. Therefore, the answer is natural habitat (C). Finish[C]

Q: Sammy wanted to go to where the people were. Where might he go? Answer Choices: (A) populated areas (B) race track (C) desert (D) apartment (E) roadblock A: The answer must be a place with many people. Populated areas, by definition, have a lot of people. Therefore, the answer is populated areas (A). Finish[A]

Q: Where do you put your grapes just before checking out? Answer Choices: (A) mouth (B) grocery cart (C) super market (D) fruit basket (E) fruit market A: The answer should be the place where grocery items are placed before checking out. Of the above choices, grocery cart makes the most sense for holding grocery items. Therefore, the answer is grocery cart (B). Finish[B]

Q: Google Maps and other highway and street GPS services have replaced what? Answer Choices: (A) united states (B) mexico (C) countryside (D) atlas (E) oceans A: The answer must be something that used to do what Google Maps and GPS services do, which is give directions. Atlases were also used to give directions. Therefore, the answer is atlas (D). Finish[D]

Q: Before getting a divorce, what did the wife feel who was doing all the work? Answer Choices: (A) harder (B) anguish (C) bitterness (D) tears (E) sadness A: The answer should be a feeling which would cause someone who was doing all the work to get divorced. If someone feels bitter towards their spouse, they are likely to want a divorce. Therefore, the answer is bitterness (C). Finish[C]

GSM8K prompt:

Q: Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May? A: Natalia sold 48/2 = «48/2=24»24 clips in May. Natalia sold 48+24 = «48+24=72»72 clips altogether in April and May. Finish[72]

Q: Betty is saving money for a new wallet which costs $100. Betty has only half of the money she needs. Her parents decided to give her $15 for that purpose, and her grandparents twice as much as her parents. How much more money does Betty need to buy the wallet? A: In the beginning, Betty has only 100 / 2 = $«100/2=50»50. Betty's grandparents gave her 15 * 2 = $«15*2=30»30. This means, Betty needs 100 - 50 - 30 - 15 = $«100-50-30-15=5»5 more. Finish[5]

Q: Julie is reading a 120-page book. Yesterday, she was able to read 12 pages and today, she read twice as many pages as yesterday. If she wants to read half of the remaining pages tomorrow, how many pages should she read? A: Maila read 12 x 2 = «12 *2=24»24 pages today. So she was able to read a total of 12 + 24 = «12+24=36»36 pages since yesterday. There are 120 - 36 = «120-36=84»84 pages left to be read. Since she wants to read half of the remaining pages tomorrow, then she should read 84/2 = «84/2=42»42 pages. Finish[42]

Q: Mark has a garden with flowers. He planted plants of three different colors in it. Ten of them are yellow, and there are 80% more of those in purple. There are only 25% as many green flowers as there are yellow and purple flowers. How many flowers does Mark have in his garden? A: There are 80/100 * 10 = «80/100 *10=8»8 more purple flowers than yellow flowers. So in Mark's garden, there are 10 + 8 = «10+8=18»18 purple flowers. Purple and yellow flowers sum up to 10 + 18 = «10+18=28»28 flowers. That means in Mark's garden there are 25/100 * 28 = «25/100 *28=7»7 green flowers. So in total Mark has 28 + 7 = «28+7=35»35 plants in his garden. Finish[35]

Q: Alexis is applying for a new job and bought a new set of business clothes to wear to the interview. She went to a department store with a budget of$200 and spent $30 on a button-up shirt, $46 on suit pants, $38 on a suit coat, $11 on socks, and $18 on a belt. She also purchased a pair of shoes, but lost the receipt for them. She has $16 left from her budget. How much did Alexis pay for the shoes? A: Let S be the amount Alexis paid for the shoes. She spent S + 30 + 46 + 38 + 11 + 18 = S + «+30+46+38+11+18=143»143. She used all but $16 of her budget, so S + 143 = 200 - 16 = 184. Thus, Alexis paid S = 184 - 143 = $«184-143=41»41 for the shoes. Finish[41]

Q: Tina makes $18.00 an hour. If she works more than 8 hours per shift, she is eligible for overtime, which is paid by your hourly wage + 1/2 your hourly wage. If she works 10 hours every day for 5 days, how much money does she make? A: She works 8 hours a day for $18 per hour so she makes 8 *18 = $«8*18=144.00»144.00 per 8-hour shift She works 10 hours a day and anything over 8 hours is eligible for overtime, so she gets 10-8 = «10-8=2»2 hours of overtime Overtime is calculated as time and a half so and she makes $18/hour so her overtime pay is 18 *.5 = $«18 *.5=9.00»9.00 Her overtime pay is 18+9 = $«18+9=27.00»27.00 Her base pay is $144.00 per 8-hour shift and she works 5 days and makes 5 * $144 = $«144 *5=720.00»720.00 Her overtime pay is $27.00 per hour and she works 2 hours of overtime per day and makes 27 *2 = $«27*2=54.00»54.00 in overtime pay 2 hours of overtime pay for 5 days means she makes 54 *5 = $270.00 In 5 days her base pay is $720.00 and she makes $270.00 in overtime pay so she makes $720 + $270 = $«720+270=990.00»990.00 Finish[990]

B.2 Training Setup

Supervised Finetuning (SFT)For SFT, we used the following hyperparameters unless otherwise stated:

learning rate	5e-6
LR warmup ratio	0.1
LR scheduler type	constant
batch size	8
batch length	1024
weight decay	0.01

Table 4: SFT hyperparameters

PPO For PPO, we used the following hyperparameters unless otherwise stated:

learning rate	1e-5
LoRA r	16
LoRA	16
LoRA dropout	0
clipping	0.2
number of rollouts	128
batch size	32
minibatch size	4
	0.95
	1
KL target	6
initial KL coef cient	0.05
training temperature	0.7

Table 5: PPO hyperparameters

C CSFT Details

C.1 Balancing Positive and Negative Samples

	positive:negative ratio	GPT-J (6B) CSQA	GPT-J (6B) GSM8K	Llama2 (7B) CSQA	Llama2 (7B) GSM8K
SFT- lter	-	47%	3%	70%	17%
CSFT	100:1	45%	3%	71%	18%
	10:1	47%	4%	69%	18%
	1:1	35%	4%	66%	15%

Table 6: CSFT with different proportions of positive and negative samples. As we increase the proportion of negative samples, performance generally decreases. At best, we only see very marginal gains using CSFT.

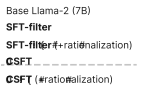
C.2 Synthetic Negatives Details

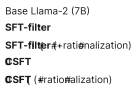
Error	Sample Trace
No errors SFT- lter	A: Tony got twice $1750 which is 2 ∗$1750 = $«2 ∗1750=3500»3500. The total amount shared was $1750+$3500 =$«1750+3500=5250»5250. #### 5250 Finish[5250]
No errors CSFT	<|good|> A: Tony got twice $1750 which is 2 ∗$1750 = $«2 ∗1750=3500»3500. The total amount shared was $1750+$3500 =$«1750+3500=5250»5250. #### 5250 Finish[5250]
1 step error CSFT	<|bad|> A: Tony got twice $1750 which is 2 ∗$1750 = $«2 ∗1750=3500»3500. The total amount shared was $1750+$3500 =$«1750+3500=5250»5250. #### 3500 Finish[3500]
2 steps error CSFT	<|bad|> A: Tony got twice $1750 which is 2 ∗$1750 = $«2 ∗1750=3500»3500. remove step #### 3500 Finish[3500]

Table 7: Sample data generated for synthetic negatives, used for netuning in Table 2. Text in red shows changes from the condition in the preceding line. Question: Q: Mr. Sam shared a certain amount of money between his two sons, Ken and Tony. If Ken got $1750, and Tony got twice as much as Ken, how much was the money shared?

D Additional Multi-Iteration Experiments

D.1 Applying CSFT





(a) Y-axis shows test set accuracy on CSQA.

(b) Y-axis shows test set accuracy on GSM8K.

Figure 3: Finetuning GPT-J (6B) with iterative **SFT-filter** and **CSFT**. In either case, we see **CSFT** with generated negatives hurts more than just training on positive samples. Note that (+rationalization) means we apply the rationalization method from STaR, where we generate prompt the model with the ground truth answer to generate reasoning in hindsight.

First, in Figure 3, we look at applying conditional training to STaR. Specifically, we follow the schedule proposed in STaR and finetune the base model from scratch at each iteration and use a fixed schedule of 40 updates initially and increasing it by 20% after each iteration, matching the batch size as proposed in the original paper. We find that rationalization is helpful for CSQA but less so for GSM8K, where the model can learn to mimic the correct final answer even with incorrect rationales. In either task, with or without rationalization, we find that **CSFT** hurts performance.