

Extended Abstract

Motivation One of the most important recent advances in AI for domains such as math and coding has been the development of 'reasoning' models i.e. 'large reasoning models' (LRMs). Despite the importance of this technique, the research community has a poor, non-rigorous understanding of how this 'reasoning training' (RT) changes the 'thought processes' of the LLMs used for this training, and, particularly, how general the 'reasoning' within the RT'd LLM is. A better understanding of reasoning models in these regards would allow us to better predict and understand the performance and generality of reasoning models compared to their base models.

Method We quantitatively and qualitatively analyze the effect of RT on LLMs by examining baseline, SFT, and RT'd model performance on the simple, synthetic domain of n -digit plus n -digit addition problems (e.g. "What is $1234 + 5678$?") for small sub-1B parameter, instruction-tuned LLMs. By using a simple synthetic task setting, we control the data distribution of training and evaluation datasets, easily generate "gold-standard" chain-of-thought solutions to problems, and easily interpret downstream model performance. We use a training dataset of n -digit addition problems, for $n \leq d$, and an evaluation dataset of n -digit addition problems, for $n > d$, to create varying levels of train-test distribution shift. We evaluate the baseline model, the model after SFT on answers to the training dataset, the model after SFT on chain-of-thought answers to the training dataset, and the model after GRPO with a normalized edit distance reward function (an accuracy-oriented metric with more resolution than a binary correct/incorrect; see 1). We also test cold-start GRPO, with an initial SFT phase on chain-of-thought answers followed by a phase of GRPO on the rest of the training dataset.

Implementation For our experiments, we use two baseline models: SmolLMv2-135M-Instruct and SmolLMv2-360M-Instruct (Allal et al., 2025). These models are pre-trained on 2 and 4 trillion tokens, respectively, and then further instruction-tuned on conversational, math, and coding data. For our experiments with SmolLMv2-135M-Instruct, we use a training dataset of approximately 15k unique n -digit addition problems, for $1 \leq n \leq 4$; we evaluate models on 10k unique n -digit addition problems, for $3 \leq n \leq 7$. For our experiments with SmolLMv2-360M-Instruct, we use a training dataset of approximately 18k unique n -digit addition problems, for $1 \leq n \leq 10$. We evaluate models on 8k unique n -digit addition problems, for $5 \leq n \leq 20$.

Results For both models, SFT on answers performed best or nearly the best in-distribution, and best out-of-distribution by a large margin, measured by both exact match accuracy and average character edit distance. SFT on chain of thought performed nearly the best or the best in-distribution, but had a steep drop-off in accuracy when moving out of distribution for both models and on both metrics. Full-RL GRPO yielded moderate gains in in-distribution accuracy and overall edit distance for the 135M model, but no accuracy boost and an edit distance penalty for the 360M model (likely due to the model tending toward degenerate answers). Cold-start GRPO performance mostly matched SFT performance for the 135M model, but interestingly provided a *unique* generalization advantage over full SFT (on chain of thought) in the 360M model when evaluated on average edit distance.

Discussion Model scale and, thus, capacity and "base capabilities", is a critical factor in determining the efficacy of particular post-training techniques, as unique generalization benefits from (cold-start) GRPO only begin to emerge with the use of a 360M parameter baseline model, and both models are unable to learn longhand addition (i.e. "gold standard chain of thought") that generalizes well even 1 digit out of distribution (and, conversely, perform significantly better when training on just problem answers). This is likely attributable to the "learning situation" faced by the model, as the fairly narrow/non-diverse dataset and small model capacity likely privilege the effective learning of the "heuristic internal addition" algorithm rather than the "step-by-step reasoning addition" algorithm. Finally, the unique advantages of GRPO may arise because of (1) the contrastive information provided by group-normalized advantage; and (2) its ability to leverage the model's existing "cognition" rather than trying to force its "thought" patterns to fit human-interpretable and -preferable ones.

Conclusion In the simple synthetic setting of n -digit addition with train-test distribution mismatch, we find that SFT on problem answers unambiguously leads to the best generalization behavior for 125M and 360M parameter LLMs fine-tuned for the task—but that cold-start (on CoT) GRPO may provide unique generalization advantages relative to SFT when paired with larger baseline models.

To N Equals Infinity and Beyond: Generalization Trends in Post-Trained LLMs

Sudharsan Sundar

Department of Computer Science
Stanford University
sjsundar@stanford.edu

Abstract

One of the most important recent advances in AI for domains such as math and coding has been the development of 'reasoning' models i.e. 'large reasoning models' (LRMs). Despite the importance of this technique, the research community has a poor, non-rigorous understanding of how this 'reasoning training' (RT) changes the 'thought processes' of the LLMs used for this training, and, particularly, how general the 'reasoning' within the RT'd LLM is. To address this question, we analyze the generalization performance of various post-training techniques, including RT via GRPO. We find that, in the simple synthetic setting of n -digit addition, evaluating on various levels of train-test distribution mismatch reveals SFT on problem answers to unambiguously lead to the best generalization behavior for 125M and 360M parameter LLMs fine-tuned for the task. SFT on chain of thought, though matching or surpassing SFT on answers on in-distribution evaluation problems, shows markedly worse generalization as evaluation problems shift to larger numbers of digits. This trend holds both when using traditional metrics such as exact match accuracy and "partial credit" metrics such as character edit distance. Cold-start GRPO appears to add little performance benefit over the initial SFT phase for our smallest 135M model, but presents a noticeable generalization advantage over SFT with our larger 360M. These results suggest the importance of inherent model capacity in making use of the power of RT via GRPO, as well as the potential for unique generalization advantages of GRPO—even compared to directly fine-tuning on "gold standard" answers—for logical/mathematical reasoning.

1 Introduction

One of the most important recent fundamental advances in ML/AI for domains with simple, verifiable answers, such as math and coding, has been the development of 'reasoning' models (sometimes referred to as 'large reasoning models' (LRMs)). These models, which are trained via RL (often GRPO (Shao et al., 2024) in particular) to produce 'reasoning traces' i.e. chains-of-thought (Wei et al., 2023) that allow them to reason through difficult problems step by step and arrive at the correct answer to a problem, have demonstrated significant increases in performance on benchmarks such as AIME 2024, Codeforces, GPQA Diamond, and MMLU, achieving or surpassing standards of respectable human performance in tasks ranging from competitive programming to competitive mathematical problems solving (DeepSeek-AI and et. al., 2025).

Despite the importance of this technique, the research community has a poor, non-rigorous understanding of how this 'reasoning training' (RT) changes the 'thought processes' i.e. inner information processing patterns of the LLMs used for this training, and what the downstream consequences are for a model's performance. In particular, an important open question is how general the 'reasoning' within the RT'd LLM is.

A better understanding of reasoning models in these regards would allow us to better predict and understand the performance and generality of reasoning models compared to their base models, how predictive 'in-distribution' benchmark performance is for similar but out-of-training-distribution (e.g. 'real-world') tasks, and so on.

To that end, we quantitatively and qualitatively analyze the effect of RT on LLMs by examining baseline, SFT, and RT'd model performance on the simple, synthetic domain of n -digit addition for small sub-1B parameter, instruction-tuned LLMs. By using a simple synthetic task setting, we are able to control the data distribution of training and evaluation datasets, easily generate "gold-standard" chain-of-thought solutions to problems, and easily interpret downstream model performance. Furthermore, a benefit of using smaller LLMs is that the task-complexity to model-capacity ratio is similar to the ratio between frontier benchmarks and frontier models, thereby providing a scaled-down analogy for the 'practical', full-scale phenomenon. For example, our baseline 360M parameter model struggles with 5-digit addition problems (20% accuracy) and completely fails on 10-digit or more addition problems (0% accuracy), providing us unsaturated and varied measures of model learning at different levels of relative difficulty.

For our experiments, we use a training dataset of n -digit addition problems, for $n \leq d$, and an evaluation dataset of n -digit addition problems, for $n > d$, to create varying levels of train-test distribution shift. We evaluate the baseline model (either a 135M or 360M instruct-tuned LLM (Allal et al., 2025)), the model after SFT on answers to the training dataset, the model after SFT on chain-of-thought answers to the training dataset, and the model after GRPO with an edit-distance-based reward function (an accuracy-oriented metric with more resolution than a binary correct/incorrect—see Section 3). We also test cold-start GRPO, with an initial SFT phase on chain-of-thought answers followed by a phase of GRPO on the rest of the training dataset.

We find that, for both models, SFT on answers performed best or nearly the best in-distribution, and best out-of-distribution by a large margin, measured by both exact match accuracy and average character edit distance. SFT on chain of thought performed nearly the best or the best in-distribution, but had a steep drop-off in accuracy when moving out of distribution for both models and on both metrics. Full-RL GRPO yielded moderate gains in in-distribution accuracy and overall edit distance for the 135M model, but no accuracy boost and an edit distance penalty for the 360M model (likely due to the model tending toward degenerate answers). Cold-start GRPO performance mostly matched SFT performance for the 135M model, but interestingly provided a *unique* generalization advantage over full SFT (on chain of thought) in the 360M model when evaluated on average edit distance.

2 Related Work

Our work is directly founded on the development of RT techniques for LLMs, particularly with the use of GRPO. As such, our work makes direct use of the introduction of GRPO (Shao et al., 2024) as well as open-source directions for training LLMs to reason in easily verifiable domains, particularly automatic accuracy-based reward functions for mathematics (DeepSeek-AI and et. al., 2025; Face, 2025). However, these works do not provide robust analyses of the generalization behavior of RT'd model performance *relative to their training data* (without controlling for this factor, we only gain a clear understanding of a model's competency in a particular area, rather than its ability to handle novel problems (Chollet, 2019)).

Our work also adds to the nascent literature studying the effects of reasoning training with rigor, including but not limited to DeepSeek-AI and et. al. (2025); Yue et al. (2025); Shojaei*† et al. (2025) (the last of which is concurrent work). Particularly, DeepSeek-AI and et. al. (2025) introduces the notion that pure-RL RT may be able to instill rare or entirely new capabilities in an LLM, as evinced by their demonstration of the "aha" moment Deepseek-R1-Zero experienced when appearing to learn the problem-solving tactic of back-tracking. However, Yue et al. (2025) challenge this perspective, and assert that RT makes a model more sample efficient, but doesn't fundamentally introduce "new" reasoning capabilities in the model. They support this analysis by showing a baseline LLM's, say, "best of 60 samples" answer is often just as good as a RT'd model's "best of 60 samples" answer, although the RT'd model's, say, "best of 5" answer is much better than the baseline LLM's. Although these two perspectives are arguably on the same spectrum¹, the latter slots nicely with

¹If A model has different "best of infinity samples" answers before and after training, then we can surely say that the model is endowed with "entirely new capabilities", but this is "monkeys typing Shakespeare" levels of

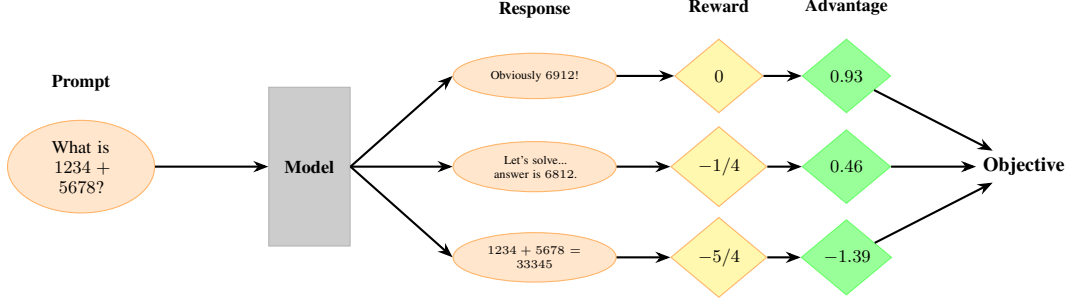


Figure 1: **Illustration of GRPO with an edit-distance-based reward.** Advantage is calculated by normalizing rewards across completions (Sagar, 2025).

the mechanics of GRPO, which derives training signal by "upweighting" the likelihood of the most relatively rewarded model completions (Shao et al., 2024) (see Figure 1 for an illustration of the GRPO process in our setting). However, neither of these works offer a rigorous analysis of how model performance changes as models are evaluated on progressively out-of-distribution tasks and, thus, don't shed light on the robustness of the reasoning learned by models, nor how it compares to other traditional post-training techniques in efficacy. Shojaei*† et al. (2025) actually adopt a fairly similar synthetic setting with a similar motivation i.e. for the same reasons we do, but their experiments only use closed-source frontier models and focus only on the direct character of the reasoning traces of models, without a comparison to other important and alternative post-training techniques. As such, Shojaei*† et al. (2025) does not account for the degree of train-test overlap for each model, nor elucidate whether there is anything special about RT relative to traditional post-training methods.

Finally, our use of a well-defined, synthetic setting for studying model generalization draws on the precedent of other such works with LLMs Garg et al. (2023); Power et al. (2022). In particular, our setting is a close analogy of Garg et al. (2023), where transformer (i.e. LLM) in-context generalization is studied in the domain D^n multiplication by training on one distribution of problems and evaluating on a somewhat differing distribution of problems. However, critically, these works do not analyze the model's capacity to use chains of thought for reasoning, compare post-training techniques, use pre-trained models, or evaluate GRPO-based RT, which limit the transferability of their results to current frontier reasoning models.

3 Method

Our task setting consists of simple n -digit plus n -digit addition problems, such as "What is 1234 + 5678?"; problems are randomly generated, and all problems within the train and eval set are unique, and each dataset is also distinct from the another. We create train-test distribution shift by varying the number of digits in problems seen during training vs. during evaluation.

Next, given a baseline LLM, we post-train the model using various techniques:

Supervised fine-tuning on correct answers. This involves typical cross-entropy-based supervised fine-tuning. The target sequences we use provide the answer to the given equation. For example, given a problem "What is 1234 + 5678?", we set the target to:

`<reasoning>1234 + 5678 = 6912<\reasoning><answer>6912<\answer>`

Supervised fine-tuning on chains-of-thought for correct answers. As above, this also involves typical cross-entropy-based supervised fine-tuning. However, the target sequences we use provide the "gold standard" chain of thought reasoning to calculate the answer (the "grade-school long-addition

sampling, and probably not practically important. Hence, what the relevant "best of n samples" is to draw the line of "fundamentally new capabilities" is an interesting one. Personally, if a baseline model's best of 100k samples is the same as the RT'd model's best of 10 samples, then that would be "entirely new capabilities" level of impressive to me.

algorithm") to the given equation. For example, given a problem "What is $1234 + 5678$?", we set the target to:

```
<reasoning>
This is a simple addition problem. Let's solve it step by step...
Step 1: 4 (digit from 1234) + 8 (digit from 5678) = 12. Write 2, carry 1.
Step 2: ...
Thus, combining our results from each step, we have  $1234 + 5678 = 6912$ .
<\reasoning>
<answer>
6912
<\answer>
```

GRPO (i.e. full-RL). We use GRPO (Shao et al., 2024) with an automated accuracy reward function. As demonstrated by Figure 1, this process involves sampling completions from the model for a given training problem, calculating the reward for each response and, from that, the advantage (i.e. reward normalized across the group of completions) of each completion. Policy updates then increase the probability of higher rewarded completions and decrease the probability of lower advantage completions. For our reward function, we use normalized character edit distance, i.e. Levenshtein distance (Yujian and Bo, 2007) divided by the length of the target string. This metric has the advantage of being continuous and therefore providing "partial credit" for partially correct answers (and more heavily penalizing very wrong answers). Thus, a model answer of "7912" vs. a correct answer of "6912" would receive a reward of $-1/4$; likewise, a model answer of "457847786912" vs. a correct answer of "6912" would receive a reward of -2 . This is particularly useful for our setting with smaller models, which, due to having lower capacity, may receive very sparse rewards during training using a binary exact match accuracy metric. Formally, we use the the GRPO objective:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E} [q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)] \quad (1)$$

$$= \frac{1}{G} \sum_{i=1}^G \left(\min \left(\frac{\pi_{\theta}(o_i|q)}{\pi_{\theta_{old}}(o_i|q)} A_i, \text{clip} \left(\frac{\pi_{\theta}(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1 - \varepsilon, 1 + \varepsilon \right) A_i \right) \right) \quad (2)$$

$$A_i = \frac{r_i - \bar{r}}{\sqrt{\frac{\sum_i (r_i - \bar{r})^2}{G}}}, \bar{r} = \frac{1}{G} \sum_i r_i \quad (3)$$

where our model parameters are θ , our old model parameters are θ_{old} , q is a sampled problem/prompt, o_i is a sampled completion for the prompt from our model, A_i is the advantage of the i th completion, and G is the total number of completions sampled. (Note: We set the KL-divergence term to 0, as it is arguably unnecessary (Hu et al., 2025).)

GRPO with cold-start data. Similar to DeepSeek-AI and et. al. (2025), we also experiment with a combined approach of an initial phase of SFT on a small portion of the training dataset, followed by a longer phase of GRPO (with an accuracy-based reward) on the rest of the training dataset.

4 Experiments & Results

For our experiments, we use two baseline models: SmoLLMv2-135M-Instruct and SmoLLMv2-360M-Instruct (Allal et al., 2025). These models are pre-trained on 2 and 4 trillion tokens, respectively, and then further instruction-tuned on conversational, math, and coding data.

4.1 Experiments with 135M Model

For our experiments with SmoLLMv2-135M-Instruct, we use a training dataset of approximately 15k unique n -digit addition problems, for $1 \leq n \leq 4$. We evaluate models on 10k unique n -digit addition problems, for $3 \leq n \leq 7$. Unfortunately, due to improper saving of the final full-RL GRPO model, we use the latest training checkpoint for our evaluations here—this model was already clearly converged at this point (nearly constant loss, small gradient update norm, and unchanging reward), so we believe this presents no meaningful obstacle to interpreting our results. See our results in Figure 2a and Figure 2b. Due to working with a smaller and easier to train model, we experiment with

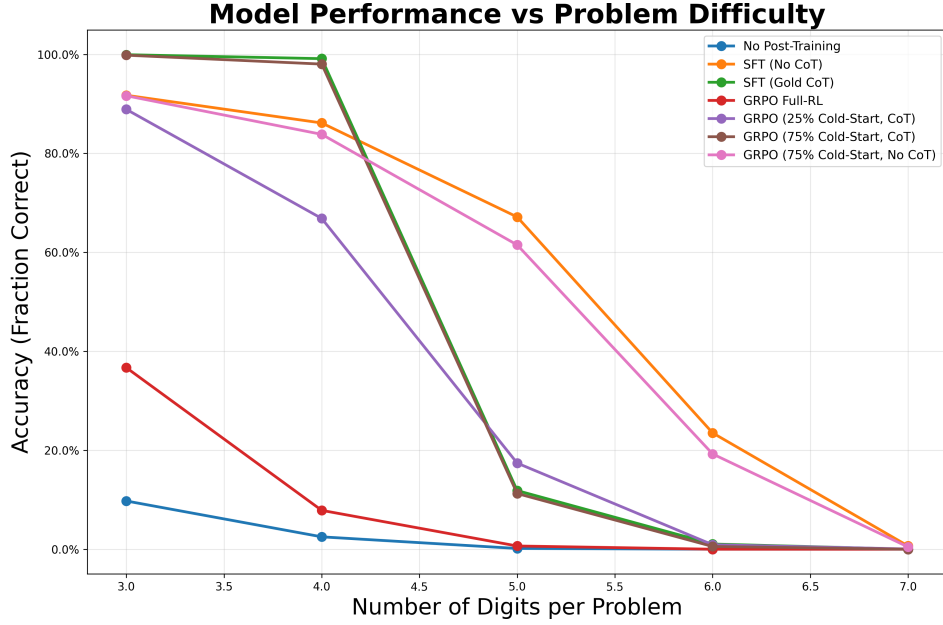
various cold start recipes: 25% SFT chain of thought \rightarrow 75% GRPO; 75% SFT chain of thought \rightarrow 25% GRPO; 75% SFT answers \rightarrow 25% GRPO. See Section 8 for further details.

Qualitative analysis of post-trained model behavior reveals a few trends. Firstly, models fine-tuned on chain of thought training data tend to learn the one-digit addition at each step of the problem-solving process, as well as the "write ones place, carry tens place" rule well, but fail to add the correct digits in the right sequence of steps when encountering an out-of-distribution training problem (see Section 9 for an example). Thus, the resulting final answer the model arrives at is wrong more often than when it simply tries to "do the addition in its head" (i.e. "reasons" via the $x + y = z$, answer-only format). Interestingly, full-RL GRPO shifts baseline model responses toward this $x + y = z$ format (rather than step-by-step reasoning), demonstrating the ability of GRPO to indicate the types of SFT training data that might be most effective.

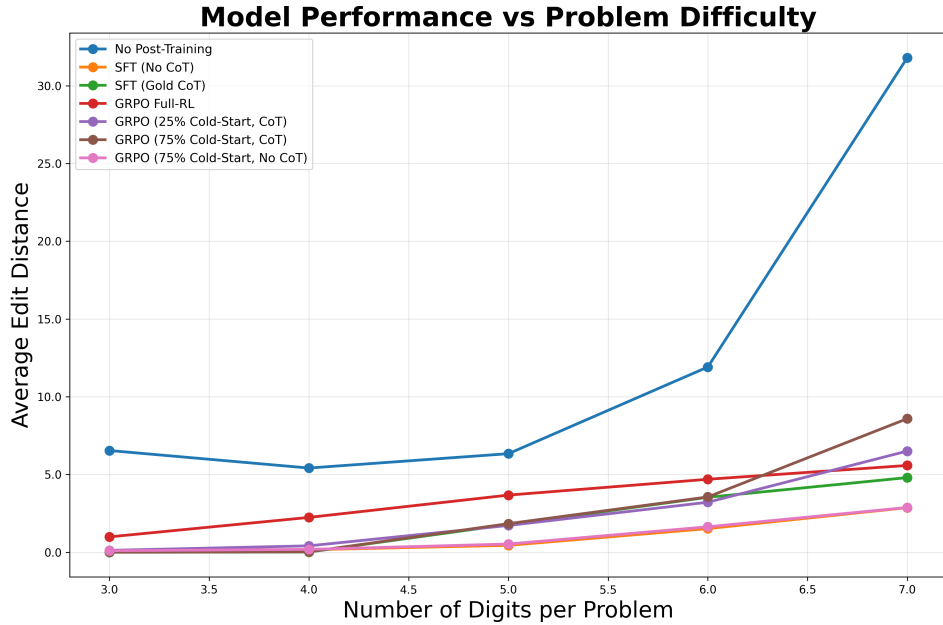
4.2 Experiments with 360M Model

For our experiments with SmolLMv2-360M-Instruct, we use a training dataset of approximately 18k unique n -digit addition problems, for $1 \leq n \leq 10$. We evaluate models on 8k unique n -digit addition problems, for $5 \leq n \leq 20$. During pure-RL GRPO, our model suffered mode-collapse and began outputting degenerate answers halfway through the training dataset. Thus, we use the last checkpoint before the model began outputting degenerate answers. See our results in Figure 3a and Figure 3b. We prioritize the use of chain of thought training data for our cold-start fine-tuning run (33% SFT on chain of thought \rightarrow 67% GRPO), as it is often the style of cold-start data used before RT (DeepSeek-AI and et. al., 2025), and training this larger model was more compute and time intensive than the 135M parameter model, so we couldn't experiment with as many cold-start configurations. See Section 8 for further details.

Qualitative analysis of model performance reveals a few interesting observations. Full-RL GRPO, when answering questions with a large number of digits (e.g. $n > 7$), degenerates to answers of approximately the correct length, but of mostly a single digit (e.g. 98888888; see Section 9 for an example), and, on occasion, gets stuck in a highly degenerate answer (e.g. 8 repeating 1000 times). This is likely what drives its edit distance up higher than the default model, as, during training, it likely learned to exploit the reward of getting a single digit right and not adding unnecessary digits to the answer (when the model fully degenerated, it earned a constant reward of -1 , which means every digit was wrong, but there were only as many digits in the answer as there should be). As with our smaller 135M model, the 360M model was also unable to correctly learn to add the digits of the two summands in correct order, even though it correctly learned the operations *within* each step of longhand addition (see Section 9 for an example). Finally, investigate the benefit of cold-start GRPO over full SFT on chain of thought by comparing the model after full SFT, only 33% SFT (i.e. before the GRPO phase begins), and the cold-start GRPO model (i.e. the model after both SFT and GRPO phases). We find that, indeed, GRPO improves on both these techniques when evaluated on character edit distance (incidentally, this analysis reveals that the full-SFT approach is likely overfit to the training dataset); see Figures 4a and 4b for details.

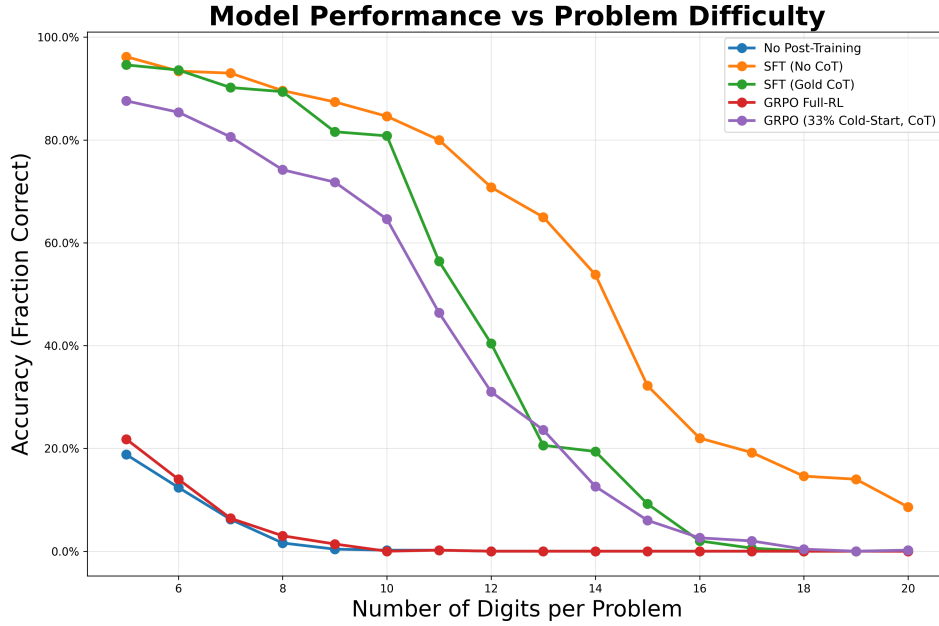


(a) Performance of various post-training techniques for SmolLMv2-135M-Instruct, exact match accuracy.

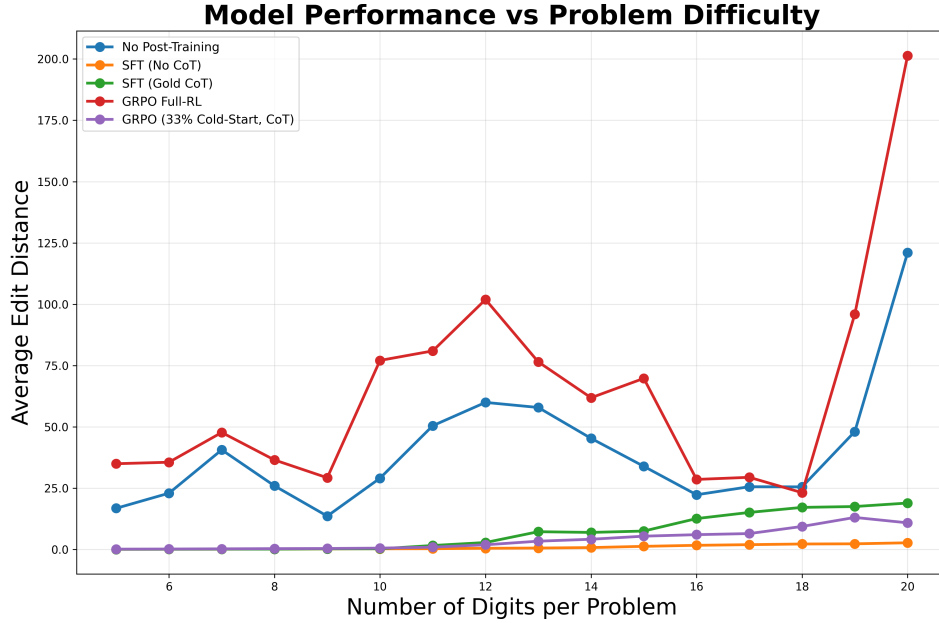


(b) Performance of various post-training techniques for SmolLMv2-135M-Instruct, average character edit distance.

Figure 2: **(Top)** SFT on chain of thought performance in-distribution (3-4 digits) is nearly perfect, but does not scale out of distribution (5-7 digits). SFT without chain of thought generalizes best out of distribution (including when used for cold-start data). 2k unique addition problems used for each digit. **(Bottom)** SFT w/ chain of thought and cold-start GRPO w/o chain of thought data perform best. Still, full-RL GRPO presents the flattest curve, suggesting surprisingly strong generalization versus other methods. 2k unique addition problems used for each digit.

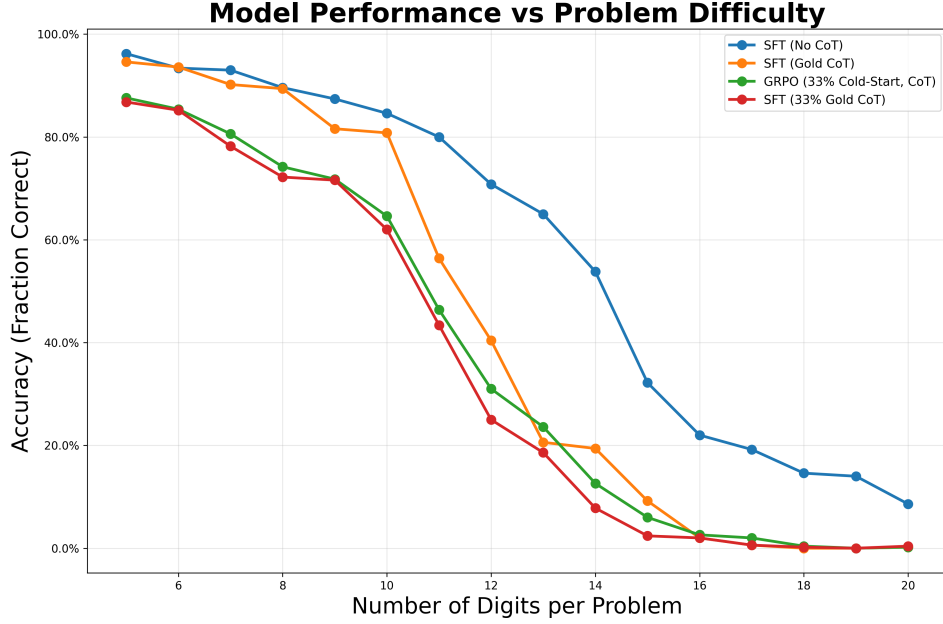


(a) Performance of various post-training techniques for SmoLMv2-360M-Instruct, exact match accuracy.

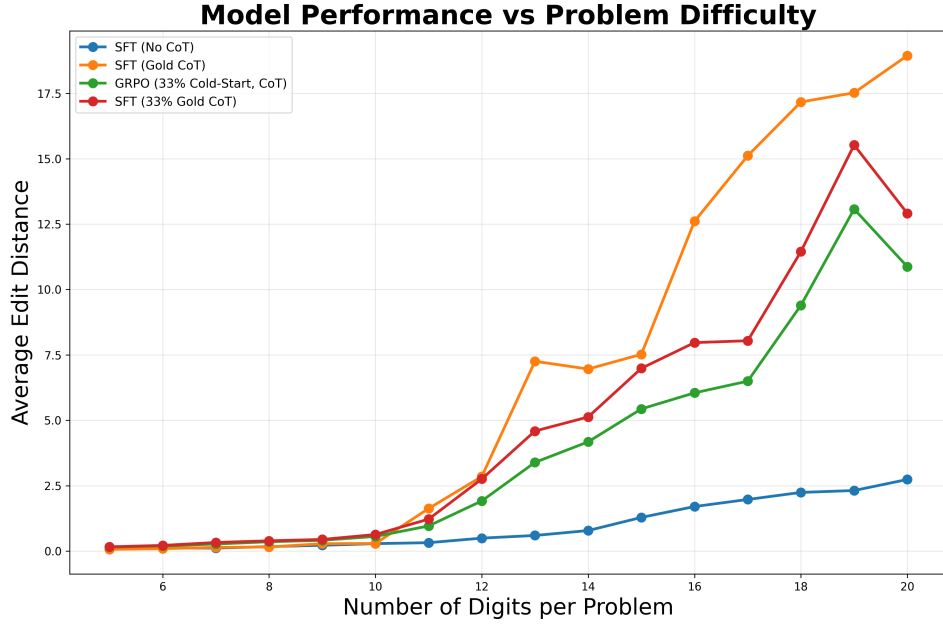


(b) Performance of various post-training techniques for SmoLMv2-360M-Instruct, average character edit distance.

Figure 3: **(Top)** SFT on answer and chain of thought perform similarly in-distribution (5-10 digits), but SFT on answers once again performs significantly better out-of-distribution (11-20 digits). Full-RL GRPO provides little to no accuracy advantage over the baseline model, likely due to proclivity to degenerate answers. Cold-start GRPO slightly under-performs full SFT on chain of thought. **(Bottom)** Cold-start GRPO confers a noticeable generalization advantage over full SFT on chain of thought. SFT on answers remains the lowest error and best generalizing technique. Full-RL GRPO actually decreases performance, likely due to tending toward degenerate answers.



(a) Performance of SFT and cold-start GRPO post-training techniques for SmolLMv2-360M-Instruct, exact match accuracy.



(b) Performance of SFT and cold-start GRPO post-training techniques for SmolLMv2-360M-Instruct, average character edit distance.

Figure 4: **(Top)** Cold-start GRPO fails to match the exact match accuracy of full SFT on chain of thought, while performing mostly similarly to 33% SFT. Thus, GRPO isn't able to match the overall correctness achieved by full SFT, particularly in- and near- distribution. **(Bottom)** Cold-start GRPO leads to noticeably less error than both full SFT and 33% SFT on chain of thought. However, SFT on answers clearly outperforms all other methods.

5 Discussion

Our results reiterate the importance of model scale: the efficacy of SFT on chain of thought is likely greatly diminished by the capacity of our under 1B parameter models—their inability to understand the correct sequence of digit additions (even just slightly out of training distribution) in fact gives the "naive" method of simply training on problem answers the best generalization performance. Furthermore, we see unique benefits to (cold-start) GRPO emerge only when using our larger 360M model.

Furthermore, our experiments with SmolLMv2-360M-Instruct demonstrate that GRPO without cold start data can be prone to collapse. But, initialized with high quality cold start data, GRPO appears to provide a *unique* generalization advantage (at least, when assigning partial credit via character edit distance) compared to SFT on even "gold-standard" chain of thought data. Perhaps because GRPO "meets the model where its at" and directs optimization in an "LLM-cognition-native" way (it only direct the model to make certain of its *own* outputs more likely, rather than, say, making a human's output more likely), as well as provides contrastive examples (e.g. elucidating the benefit of getting all digits right vs all but 1 digit right in an addition problem), whereas SFT tries to mold the model directly to a human-interpretable and preferable pattern (perhaps analogous to "trying to get a fish to climb a tree"). This interpretation squares with our observation and general "folk wisdom" that RT's efficacy is highly dependent on the base model's capacity and competency.

In addition, our analysis reinforces the conclusion that SFT is a formidable technique that, although simple, can often get the job done quite well, especially with smaller models that benefit from being "spoon-fed" (i.e. teacher-forced) information on how to solve problems and the correct answers to problems.

Finally, it is interesting to see the failure of chain of thought to out-perform generalization of simply training on problem answers, *even* when using a continuous evaluation metric based on the number of digits differing between the model and correct answer. Our intuition here is that learning step-by-step reasoning requires a much higher "fixed" complexity/learning cost, but, once this ability is acquired, it can succeed at arbitrary problems (i.e. has very low "variable", per-example learning cost). The opposite seems true for direct answer prediction, since, for example, it is much simpler to memorize addition heuristics (i.e. "internalize" an algorithm for addition, a low "fixed" learning cost), and still generalizes relatively well (i.e. "variable" learning cost is fairly low for other addition problems). Given our training set is quite narrow (it is only composed of n -digit addition problems), and our model is low capacity, it likely privileges the learning of the narrower solution algorithm compared to more general reasoning. It seems the smaller, pre-trained and instruct-tuned models we used fit this learning paradigm.

6 Conclusion

We analyze the generalization performance of various post-training techniques. We find that, in the simple synthetic setting of n -digit addition, evaluating on various levels of train-test distribution mismatch reveals SFT on problem answers to unambiguously lead to the best generalization behavior for 125M and 360M parameter LLMs fine-tuned for the task. SFT on chain of thought, though matching or surpassing SFT on answers on in-distribution evaluation problems, shows markedly worse generalization as evaluation problems shift to larger numbers of digits and move further out of the training distribution. This trend holds both when using traditional metrics such as exact match accuracy and "partial credit" metrics such as character edit distance. Full-RL GRPO, when it doesn't lead to degenerate answers, hints at better generalization on the character edit distance metric, but its high baseline error makes it questionable whether such a trend would hold. Cold-start GRPO appears to add little performance benefit over the initial SFT phase for our smallest 135M model, but presents a noticeable generalization advantage over SFT with our larger 360M. These results suggest the importance of inherent model capacity in making use of the power of RT via GRPO, as well as the potential for unique generalization advantages of GRPO—even compared to "gold standard" answers—for logical/mathematical reasoning.

7 Team Contributions

- **Group Member 1: Sudharsan Sundar.** Everything :).

Changes from Proposal Very few changes. I unfortunately didn't have time to replicate Yue et al. (2025) or use enough of a diversity of models to construct a scaling law, but, otherwise, the core methodology and experiments have been consist (note: I ended up using addition rather than multiplication as the problem setting, since it would likely be easier for the smaller models I use).

References

- Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model. arXiv:2502.02737 [cs.CL] <https://arxiv.org/abs/2502.02737>
- François Chollet. 2019. On the Measure of Intelligence. arXiv:1911.01547 [cs.AI] <https://arxiv.org/abs/1911.01547>
- DeepSeek-AI and Daya Guo et. al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL] <https://arxiv.org/abs/2501.12948>
- Hugging Face. 2025. Open R1: A fully open reproduction of DeepSeek-R1. <https://github.com/huggingface/open-r1>
- Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. 2023. What Can Transformers Learn In-Context? A Case Study of Simple Function Classes. arXiv:2208.01066 [cs.CL] <https://arxiv.org/abs/2208.01066>
- Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025. Open-Reasoner-Zero: An Open Source Approach to Scaling Up Reinforcement Learning on the Base Model. arXiv:2503.24290 [cs.LG] <https://arxiv.org/abs/2503.24290>
- Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. 2022. Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets. arXiv:2201.02177 [cs.LG] <https://arxiv.org/abs/2201.02177>
- Yuvraj Sagar. 2025. Understanding GRPO: Key ingredient behind Deepseek-R1's Success. <https://medium.com/@yuvrajsagar117/understanding-grpo-key-ingredient-behind-deepseek-r1s-success-5f8f05494e6d>
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300 [cs.CL] <https://arxiv.org/abs/2402.03300>
- Parshin Shojaei*, Iman Mirzadeh*, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025. The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity. <https://ml-site.cdn-apple.com/papers/the-illusion-of-thinking.pdf>
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs.CL] <https://arxiv.org/abs/2201.11903>
- Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model? arXiv:2504.13837 [cs.AI] <https://arxiv.org/abs/2504.13837>
- Li Yujian and Liu Bo. 2007. A normalized Levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence* 29, 6 (2007), 1091–1095.

8 Implementation Details

Here, we details some our hyperparameter choices.

For SFT runs, for our 135M model, we use batch size 16; for our 360M model we use batch size 8. We use learning rate $1e-5$ and use all other default hyperparameters from Huggingface’s SFTConfig and SFTTrainer classes.

For GRPO runs, we use batch size 16 (prompts/question per step) and 4 generations per prompt. We use all other default hyperparameters from Huggingface’s GRPOConfig and GRPOTrainer classes (e.g. temperature is set to 1.0, beta is set to 0.0, etc.).

For evaluation, we limit generations to 2048 tokens (even 20 digit problems with gold chain of thought can be solved in under 1200 tokens); we sample from the evaluated model at temperature 0.2 and top-p 0.9.

9 Examples of Model Outputs

9.1 135M Model

9.1.1 SFT on CoT

```
"question": "211367 + 637560"
"correct_answer": "848927"
"model_response": "assistant
<reasoning>
This is a simple addition problem. Let’s solve the problem step by step by
adding the digits from right to left.

Step 1: 7 (digit from 211367) + 0 (digit from 637560) = 7. Write 7, carry 0.
Step 2: 6 (digit from 211367) + 6 (digit from 637560) + 0 (previous carry)
= 12. Write 2, carry 1.
Step 3: 1 (digit from 211367) + 3 (digit from 637560) + 1 (previous carry)
= 5. Write 5, carry 0. **INCORRECT STEP**
Step 4: 2 (digit from 211367) + 6 (digit from 637560) + 0 (previous carry)
= 8. Write 8, carry 0.

Thus, combining our results from each step, we have 211367 + 637560 = 855223
</reasoning>
<answer>
855223
</answer>"
```

9.2 360M Model

9.2.1 Full-RL GRPO

```
"question": "39339559781 + 64326989377"
"correct_answer": "103666549158"
"model_response": "assistant
To add 39339559781 and 64326989377, we need to perform the addition step by step.

First, we add the hundreds digits: 3 + 6 = 9.

Next, we add the tens digits: 9 + 9 = 18.

Then, we add the ones digits: 1 + 8 = 9.

So, the final result is 99999999999."
```

9.2.2 SFT on CoT

"question": "21787248886844 + 48440677878123"

"correct_answer": "70227926764967"

"model_response": "assistant

<reasoning>

This is a simple addition problem. Let's solve the problem step by step by adding the digits from right to left.

Step 1: 8 (digit from 21787248886844) + 3 (digit from 48440677878123) = 11 . Write 1, carry 1.

Step 2: 8 (digit from 21787248886844) + 8 (digit from 48440677878123) + 1 (previous carry) = 17 . Write 7, carry 1. ****INCORRECT STEP****

Step 3: 8 (digit from 21787248886844) + 7 (digit from 48440677878123) + 1 (previous carry) = 16 . Write 6, carry 1.

Step 4: 8 (digit from 21787248886844) + 7 (digit from 48440677878123) + 1 (previous carry) = 16 . Write 6, carry 1.

Step 5: 4 (digit from 21787248886844) + 7 (digit from 48440677878123) + 1 (previous carry) = 12 . Write 2, carry 1.

Step 6: 2 (digit from 21787248886844) + 0 (digit from 48440677878123) + 1 (previous carry) = 3 . Write 3, carry 0.

Step 7: 7 (digit from 21787248886844) + 4 (digit from 48440677878123) + 0 (previous carry) = 11 . Write 1, carry 1.

Step 8: 8 (digit from 21787248886844) + 4 (digit from 48440677878123) + 1 (previous carry) = 13 . Write 3, carry 1.

Step 9: 7 (digit from 21787248886844) + 8 (digit from 48440677878123) + 1 (previous carry) = 16 . Write 6, carry 1.

Step 10: 1 (digit from 21787248886844) + 4 (digit from 48440677878123) + 1 (previous carry) = 6 . Write 6, carry 0.

Step 11: 2 (digit from 21787248886844) + 8 (digit from 48440677878123) + 0 (previous carry) = 10 . Write 0, carry 1.

Step 12: 1 (final carry) = 1. Write 1.

Thus, combining our results from each step, we have $21787248886844 + 48440677878123 = 60227926764967$

</reasoning>

<answer>

60227926764967

</answer>"