Reasonix: An exploration into LLM reasoning with CoT SFT and RLHF on small base models.

Edwin Chacko **Zhongyu Xu** **Zicheng Wang**
University of Toronto
{edwin.chacko, xuzhong8, zicheng.wang}@mail.utoronto.ca

Abstract

Large Language Models (LLMs) have succeeded in a wide range of tasks in language modelling and information retrieval. However, their performance is limited when dealing mathematics problems that requires with logical and multi-step reasoning. To enhance LLMs' reasoning ability in mathematics, we have developed a finetuning pipeline that integrates the Llama base models with techniques including Chain of Thought (CoT), supervised fine-tuning (SFT), and reinforcement learning (RL) [1]. These processes allow models to demonstrate their thinking process (CoT) before providing the final answers, using a basis of reasoning before answering to avoid next-token prediction guesswork. We applied our methods to Llama 3.2 models with a focus on Llama 3.2 3B, and avoiding Low-Rank Approximation (LoRA) training. As a result, our CoT SFT model's performance on the GSM8K dataset improved significantly from 17% to 54%, compared to the baseline, and the proportion of completely failed reasoning responses decreased from 31% to 18%.

1 Introduction

The field of reasoning language models itself is state-of-the-art, with top models such as OpenAI's o1 being proprietary and not open-source [12]. Language models are token predictors; initially, they are not built for reasoning. In light of this, we introduce **Reasonix**, an exploration into base language models' reasoning enhancement. During the planning stage, we dived into the developments of various reasoning-enhanced LLMs, including Kimi K1.5 [16], TULU 3 [10], and DeepSeek R1 [5]. The commonalities that we found in the training processes of these models highlight the effectiveness of using CoT SFT and RL to promote reasoning. However, our goal is not to surpass the performance of these models, but to demonstrate that our pipeline effectively enhances the reasoning performance of base models.

A language model's reasoning ability can be demonstrated across diverse fields such as math problem-solving, coding, logical inferencing, etc. We choose to focus on assessing the model's math problem-solving ability as our core benchmark for evaluation. Math problems often involve multiple steps, complex relationships, and the application of various concepts. This complexity can serve as a rigorous test of the model's ability to understand and manipulate information, revealing its depth of reasoning. In most cases, they also provide clear right or wrong answers, allowing for straightforward evaluation of a model's performance. The baseline models we used are LlaMA 3.2 Base (1B and 3B) [7], and the benchmark accuracies are 3.2% and 16.80%, respectively, evaluated using GSM8K, a grade school math problem dataset that contains around 8,800 data samples [4]. This poor performance in solving simple math problems is because LlaMA 3.2 base models are base transformer decoder models that are only pre-trained for next-token prediction [17]. These raw pre-trained models rely on the self-attention mechanism to perform simple tasks such as predicting words, detecting patterns, and generating coherent but unverified text, as illustrated by Figure 9 in Appendix A.2. Without instruction fine-tuning, they struggle to follow instructions, and they lack CoT reasoning

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

when trying to solve logic-based problems. The limitations of these models underscore the need for an efficient training process to foster a more mature reasoning ability.

Figure 1 decribes the Reasonix development pipeline, which begins by few-shot prompting the base models, Llama 3.2 Base 1B & 3B, to recognize the pattern to solve problems step by step. Next, Llama 3.2 3B-Instruct is used to generate CoT data samples from the dataset that contains only questions and final answers; these CoT data are used for CoT fine-tuning. Finally, we apply reinforcement learning with human feedback to create our final model.

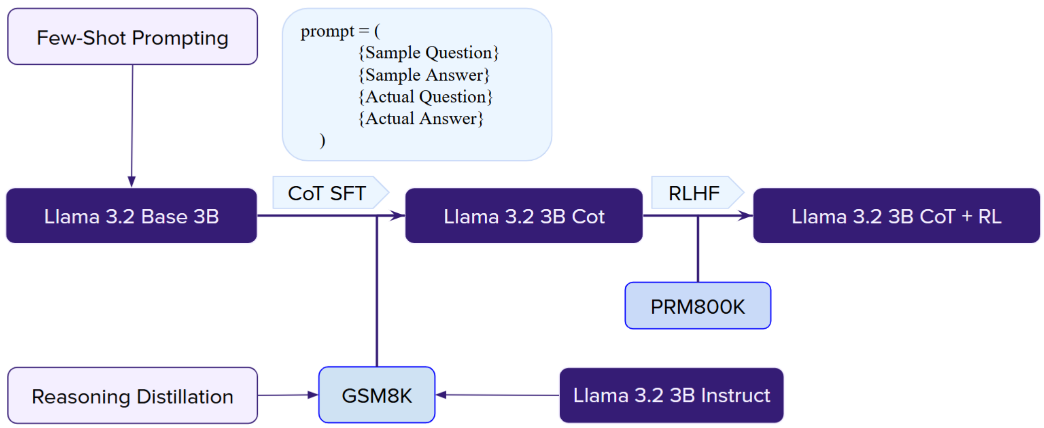

Figure 1: The Reasonix development pipeline including, distillation, CoT SFT, RLHF.

2 Methodology

2.1 Dataset and Pre-Processing

As previously mentioned, the Llama 3.2 base models are incapable of understanding the instructions of the prompts; this is shown by their poor performance in following guidelines and solving simple math problems. Therefore, we apply a few-shot prompting technique before the evaluation of base models [3]; as illustrated in Figure 1, by feeding the model with a predefined question-solution sample pair, the model can quickly generalize a systematic approach for problem-solving before it starts processing the next real task. In result, the model becomes less likely to generate irrelevant or non-sensical responses.

GSM8K is a well-annotated dataset designed by human problem writers to test the arithmetic reasoning and problem solving ability of a model [4]. It is comprised by 7,500 training data and 1,300 test data. Each sample contains a grade school math problem and a corresponding solution process that leads to the final numerical value. The math problems range from simple arithmetic to more complex multi-step questions.

GSM8K has been widely used to benchmark the ability of LLMs to solve math problems. Since many researchers use GSM8K and compare their results on its leaderboard, it allows us to easily compare the performance of our model. The training split of this dataset is used during CoT SFT stage and the test split is used for evaluation. Details on these stages are provided in the later sections.

2.2 Reasoning Distillation

One deficiency in using GSM8K for training is that, for each sample question, the dataset provides only a brief arithmetic process to the final numerical answer, without explicit reasoning steps. Since the goal is to enhance the reasoning ability of base models, we also want to provide a reasoning guideline for each sample during the training stage.

Therefore, we pre-process the dataset using a technique known as reasoning distillation or self-instruction, which focuses on transferring not just the answer, but the reasoning process from a larger model to a smaller one [15]. To perform this data augmentation technique, we feed GSM8K to Llama

3.2-3B-Instruct and prompt it to generate a higher quality step-by-step solution for each question, as illustrated in Figure 2. These chain-of-thought data are then used to train the base models during the CoT SFT stage.

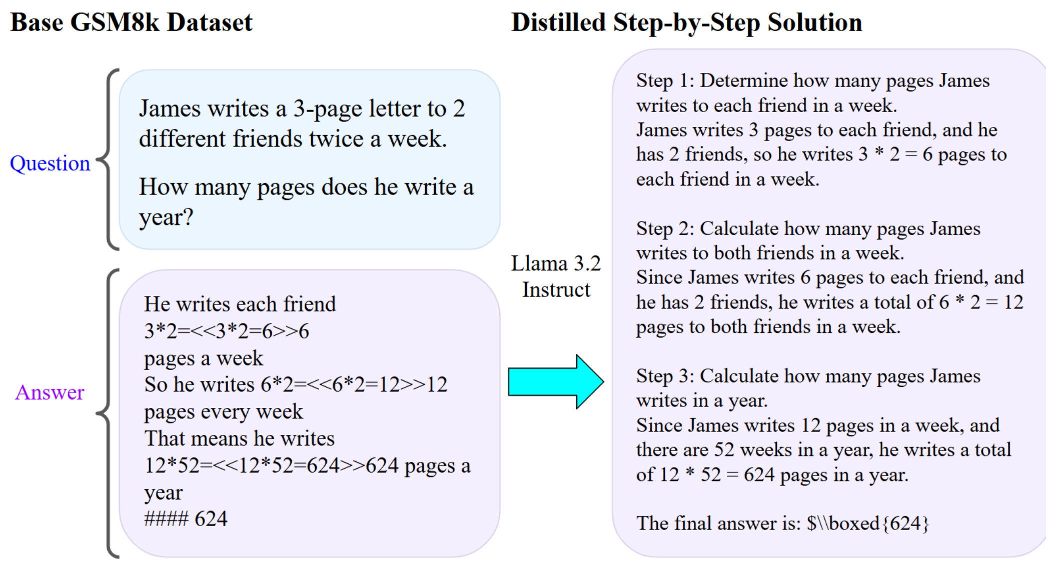

Figure 2: An illustration of distilling the reasoning process for a question from Llama 3.2 3B-Instruct.

2.3 Chain-of-Thought Supervised Fine-Tuning (CoT SFT)

CoT Supervised Fine-Tuning (CoT SFT) is a training technique where a language model is fine-tuned using chain-of-thought (CoT) reasoning examples with human-annotated or model-generated rationales; an example of this CoT data is shown in Figure 10 in Appendix A.3. It is used to improve the model's multi-step reasoning ability to increase accuracy, while introducing more transparency and interpretability to the model's output [8].

In this stage, we train the base model (Llama 3.2 1B/3B) using the previously distilled dataset. The loss is computed between the model's generated answer and the step-by-step solution to reward the model for predicting step-by-step answers. This is helps address the earlier mentioned issues in reasoning as the model is forced to break the problem into smaller steps. The rationale behind this process is to help condition the model on its previous tokens, allowing for less variability and more coherence.

2.4 Reinforcement Learning with Human Feedback (RLHF)

RLHF is a method used to align model outputs with human-preferred answer structures. This has vast application in LLMs beyond reasoning, and can be used to enable outputs that better reflect human values, tone, or style [2]. The three components of RLHF are discussed here: data, reward model, and policy model.

2.4.1 Dataset

The dataset used for RLHF training is the PRM800k dataset which consists of human labeled mathematical reasoning steps on a range of math problems varying immensely in complexity [11]. Each sample is a step in a reasoning process generated by various GPT models. The labels are in $\{-1, 0, 1\}$ corresponding to bad, okay, and good respectively, and represent the quality of each step. Quality is defined by the human annotator, but is generally helpfulness and accuracy [9]. Figure 11 in Appendix A.4 displays some example input-label pairs.

2.4.2 Reward Model

The reward model is optional but often used for modularity and scalability. The reward model is a copy of the base LLM we are interested in training, with a classification head attached. The classification head is a MLP and the outputs correspond to the labels of the dataset. This new model is trained using the Low-Rank Adaptation (LoRA) to predict the labels of the PRM800k dataset [9].

The reward model is optional as we can equivalently use the labels from the dataset itself, however a reward model allows extending the policy model training beyond the labeled dataset. The idea is that the reward model has learned the preferences of the human annotators and can thus replace them to label any output the policy model makes on any similar data [13]. This is more generalizable and standard practice.

2.4.3 Policy Model

The policy model is the final model we can use for evaluation once trained. The policy model is initialized to a pretrained LLM we are interested in aligning. We use the Proximal Policy Optimization (PPO) objective [14] which updates the policy π_θ to maximize the expected reward while staying close to the reference policy $\pi_{\theta_{\text{ref}}}$:

$$\mathcal{L}_{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right].$$

Where ϵ is the PPO clip threshold and,

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{ref}}}(a_t|s_t)}$: probability ratio between current and reference policy
- $\hat{A}_t$: advantage estimate (reward minus value function)

The use of the `min` and `clip` operations prevents large policy updates by limiting the impact of excessively high or low probability ratios. This helps maintain stability during training and keeps the policy from deviating too far from the reference model. Typically, the reference policy is a frozen copy of the initial pretrained model, ensuring that the policy retains beneficial prior knowledge while being nudged toward higher-reward outputs. In addition to PPO's clipped objective, we also include a Kullback–Leibler (KL) divergence penalty between the current policy π_θ and the reference policy $\pi_{\theta_{\text{ref}}}$. This KL term acts as an explicit regularizer to discourage the updated policy from deviating too far from the initial model.

Figure 3 outlines the pipeline for RLHF training which integrates the reward model and policy model as described in this section.

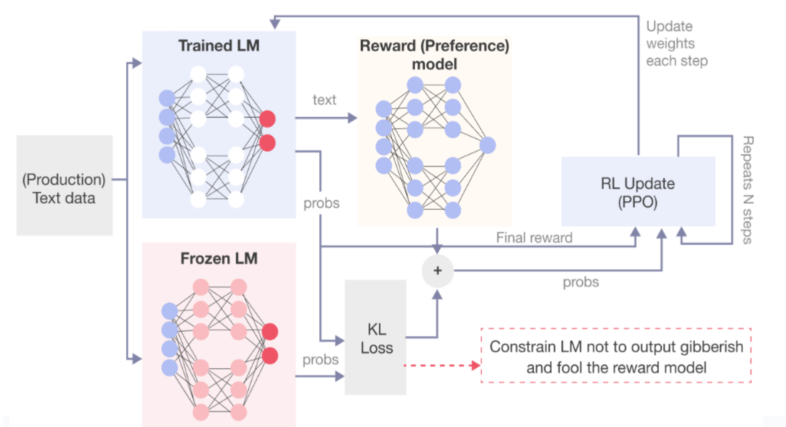

Figure 3: RLHF Training Pipeline for the policy model [6].

2.5 Training Setup and Hardware Constraints

All training was conducted using two NVIDIA A100 GPUs. For CoT baseline models, training was performed both with and without LoRA adapters, as full fine-tuning was feasible for these smaller architectures. However, the reward model and policy model were trained exclusively using LoRA, as full-parameter tuning proved computationally impractical due to the size of the base LLMs and the time required.

In addition, it is important to note that the policy model could not be trained on the PRM800k train set as this would take 250 hours on the A100 cluster and cost $500. Thus, it was decided to train the policy model on the PRM800k test set. The reward model was still trained on the PRM800k train set.

All LoRA-based models were implemented using the PEFT (Parameter-Efficient Fine-Tuning) Python library.[1] The trade-offs and implications of using LoRA for RLHF training are discussed in the following section.

Additional details about the project's hardware environment are provided in Appendix A.1.

3 Results and Discussion

3.1 Metrics

For our evaluation, we use accuracy as the primary evaluation metric. Since the GSM8K dataset contains grade school math problems where each question has a definitive final numerical answer. This allows us to perform a direct comparison between the model's predicted answer and the ground truth. In our prompt, we instruct the model to output its final numerical answer in a special format, enabling us to easily extract this numerical value from the model's response. Another important reason is that accuracy is the standard metric used in the official GSM8K benchmark leaderboard. Using the same metrics enables us to compare our model's performance directly with other state-of-the-art models.

Using accuracy as the sole metrics does not allow us to analyze in depth on why the baseline model fails to reason. Therefore, we have developed a customized evaluation metrics. We design four categories based on the types of the response provided by the model, we classify each answer into one of four categories. The four categories are defined as follows:

Category 1: Correct Final Numerical Answer with Correct Reasoning (Fully Correct)

Description: The model provides a clear and correct solution. Each step includes the correct setup for numerical evaluation, and when all steps are combined, the final numerical answer is correct.

Category 2: Solid Reasoning with Minor Arithmetic Errors (Arithmetic Error)

Description: The model gives step-by-step explanation similar to Category 1, showing correct logical reasoning for the math problem in every steps. However, the model makes small arithmetic mistakes when calculating the numerical values, leading to an incorrect final answer.

Category 3: Partially Correct Reasoning (Partial Reasoning)

Description: The model sets up a basic chain-of-thought structure and produces a final numerical answer. However, only some of the reasoning steps are accurate. Errors in one or more intermediate steps cause the overall solution to be incorrect.

Category 4: Complete Failure to Engage in Reasoning (Failed Reasoning)

Description: The model does not generate any meaningful chain-of-thought reasoning. The response might simply repeat the question, produce irrelevant content, or get stuck in a loop without making any progress toward a solution. There is no attempt to analyze or solve the problem.

3.2 Results

The resulting accuracies of all the baseline and trained models is shown Figure 4. The rightmost model is the RLHF policy model using the 3B + CoT as a starting point. The LoRA was applied to the policy model during policy training. In addition, the reward model achieved an accuracy of 68% on the PRM800k test set, which is typical and sufficient to train the policy model.

[1] https://github.com/huggingface/peft

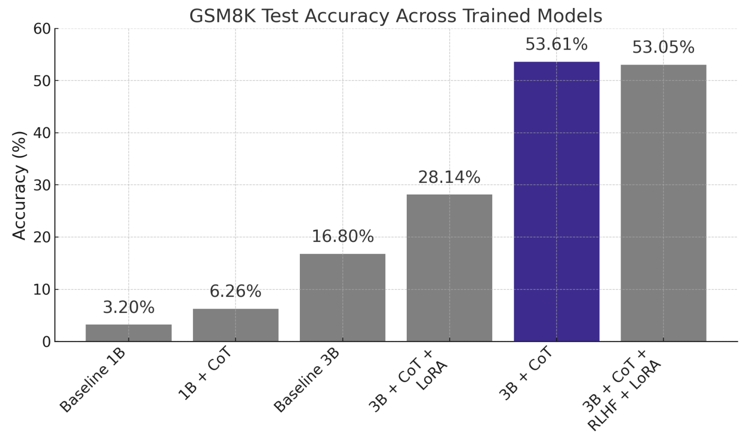

Figure 4: GSM8K Test Accuracy Across Trained Models

From the figure, we see that the 1B models perform poorly on reasoning tasks, before and after CoT. CoT-SFT helps bigger models because they're capable of handling the extra cognitive load. Smaller models can't "think step by step" because they don't have the headroom to hold all those steps. For this reason, we will limit our discussion to the 3B model.

Another noteworthy observation is the difference when training LoRA adapters versus the full model. The LoRA CoT model only sees an increase of 11% while the full model sees an increase of a staggering 37%. LoRA introduces a bottleneck in representational flexibility by updating only low-rank projections of the model's weights. This constraint may limit the model's ability to fully adapt to the multi-step reasoning structure demanded by CoT-style supervision. This is highly informative about the nature of reasoning, suggesting that more of the reasoning in smaller models occurs outside of the attention layers.

3.3 Baseline Performance

We first evaluate the accuracy of our baseline model, we modified the prompt given to our model so that it follows a similar format. The prompt instructs the model to provide a detailed explanation and to output the final numerical answer on a new line within "\boxed{}". After the model generates its response, we extract the numerical value and compare it with the ground truth. We calculate a simple binary accuracy where each prediction is marked as correct if the extracted number matches the ground truth, or incorrect otherwise. The accuaracy of our baseline model is only 16.8 percent.

To further understand why the model fails to reason correctly, we sampled a subset of the baseline model's outputs and categorized them into the four categories that we have defined. In Figure 5, we show example responses for each of the four categories. For the example response in the first category, we see the model output correctly sets up the evaluation based on its analysis of the question, and it correctly computes the final numerical answer, leading to a correct solution. For the example response in the second category, the model's output contains a correct setup of the evaluation step; however, it fails to compute the decimal multiplication accurately. As shown in the figure, it fails to evaluate the expression "9 * 1.87," resulting in an incorrect numerical answer. The example response in the third category has a complete chain-of-thought structure, but it makes an error in the reasoning steps because it fails to understand the time relationship between the faster and slower racing teams, causing the model to output a negative time. The example response in the fourth category fails to establish a complete chain-of-thought structure, as it repeatedly loops through the same step, "Joanne gathers 30 apples from the tallest trees…", and it does not provide any numerical value at the end.

We summarize the number of responses from baseline model in each category in Figure 6, 31% of the responses fall into Category 4, where the model completely fails to generate a chain-of-thought response. This outcome is expected because the baseline model is trained for conversational interactions rather than for detailed chain-of-thought reasoning to solve math problems.

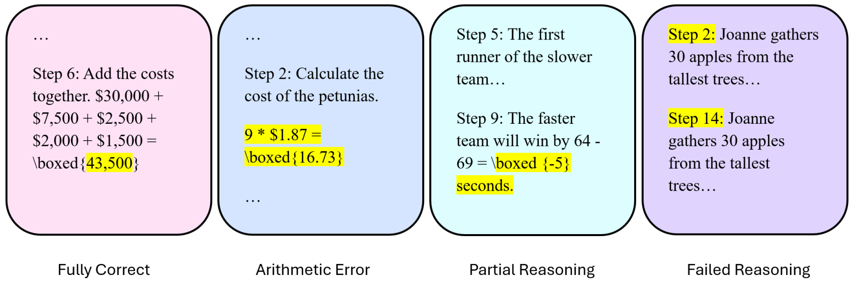

Figure 5: Examples illustrating each of the four categories for errors.

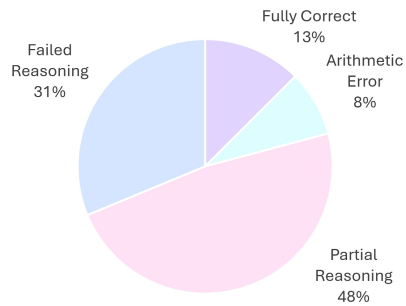

Figure 6: A pie chart of the four categories for errors.

3.4 COT finetuned Performance

We have fine-tuned the baseline model by applying our COT SFT technique outlined in Section 2.3, the accuracy increases significantly from 17 percent to 54 percent as shown in the left bars in Figure 7. This increase in performance shows that the CoT model can reason better and generate high quality, correct reasoning steps that are required to solve more complex math problems.

In Figure 7, the bars on the right shows the number of outputs in category 4. Prior to finetuning, 31 percent of the baseline model's outputs fell into this category, where the model completely failed to engage with the problem or to generate any meaningful COT style response. With our finetuned model, these failures dropped to 18 percent. This change indicates that the model is now more capable to engage with the question that the baseline model completely failed to address. The performance gains can be largely attributed to the data augmentation we have applied to the training data in GSM8K, allowing the model to learn from structured reasoning steps.

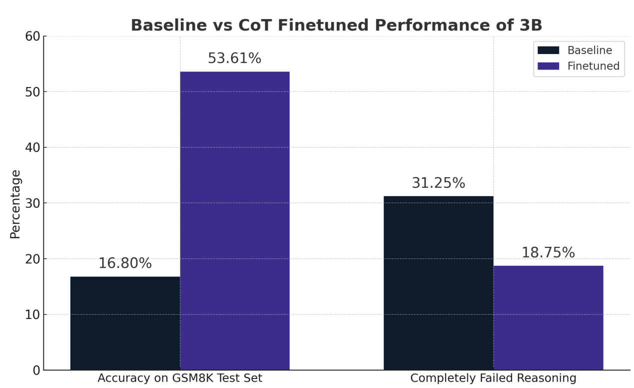

Figure 7: A histogram showcasing a significant increase in accuracy and a decrease in the number of completely failed reasoning cases.

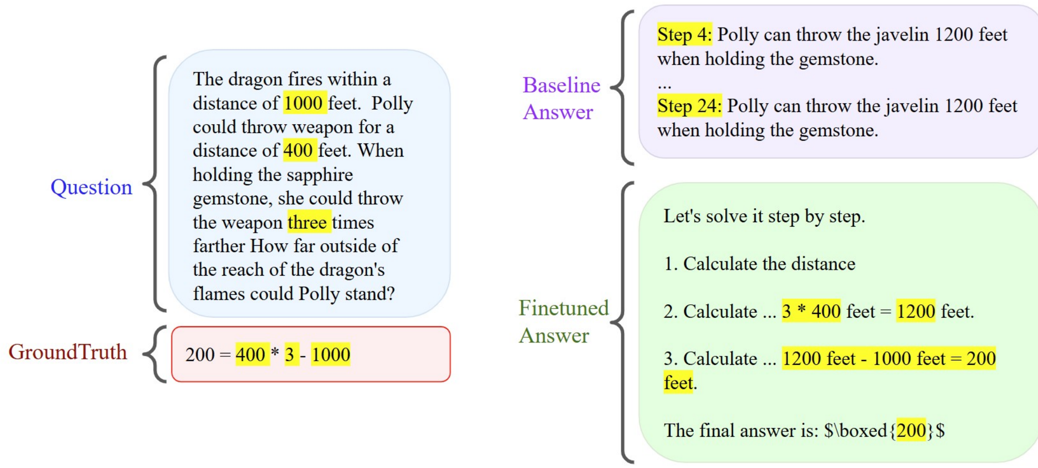

Figure 8: Comparing the model's response before and after training. The fine-tuned model is able to reason correctly on cases failed by the base model.

In addition, the quality of the model's chain-of-thought style responses also has improved. The response from our fined tuned model has become more structured, with clear and logical reasoning steps. Having a series clear reasoning steps allow us to analyze each step to interpret model's underlying logic. In Figure 8, we demonstrate with an example that our finetuned model can now reason correctly on a question that failed to be answered by the baseline model. The baseline model fails to analyze the question and generate a response that keeps repeating the same step, whereas our finetuned model correctly answer the question with clear intermediate steps.

3.5 RLHF Performance

The RLHF model performed slightly worse than the pre-policy trained (3B + CoT). This is unexpected as further alignment is expected to increase performance. As mentioned in Section 2.5, hardware limitations prevented training the policy model on the full train set and we settled for training on the test set. In addition, the policy model was trained with LoRA adapters which, as seen in Figure 4 is not optimal for reasoning.

In addition, the policy model was trained using LoRA adapters, which is not ideal for reasoning learning. This may responsible for the decrease in performance.

These limitations serve as the path forward and next steps. Training non-LoRA and on the train set will provide great insight to how RLHF compliments CoT. In addition, comparing the performance of using the base Llama 3.2 3B model for policy training will provide useful insights into the nature of each method. The same tests can also be extended to the 1B model to see if these approaches can help overcome the size limitations.

4 Conclusion

Our work focused on improving the reasoning ability of LLMs in the domain of mathematics. Our pipeline integrated chain-of-thought supervised fine-tuning and reinforcement learning with human feedback to fine-tune baseline models. We focused on Llama 3.2 3B and saw to an significant improvements in both accuracy and quality of reasoning. Specifically, our finetuned model's performance on the GSM8K dataset improved from 17% to 54%, compared to the baseline, and the proportion of completely failed reasoning responses decreased from 31% to 18%. We explored RLHF which required significant hardware requirements, preventing us from attaining meaningful analysis. This report demonstrate the effectiveness of our training pipeline, and it also show that finetuning LLMs with structured chain-of-thought data can overcome the inherent limitations of next-token prediction, enabling the models to generate responses with step-by-step reasoning rather than simply predicting the next token.

References

[1] Gianni De Fabritiis Andrew White. *AI for science with reasoning models*. 2025. URL: https://diffuse.one/p/d1-007.

[2] Yuntao Bai et al. *Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback*. 2022. arXiv: 2204.05862 [cs.CL]. URL: https://arxiv.org/abs/2204.05862.

[3] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[4] Karl Cobbe et al. *Training Verifiers to Solve Math Word Problems*. 2021. arXiv: 2110.14168 [cs.LG]. URL: https://arxiv.org/abs/2110.14168.

[5] DeepSeek-AI. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. 2025. arXiv: 2501.12948 [cs.CL]. URL: https://arxiv.org/abs/2501.12948.

[6] H. Dhaduk. *What is Reinforcement Learning from Human Feedback (RLHF)? Benefits, Challenges, Key Components, Working*. 2023. URL: https://www.simform.com/blog/reinforcement-learning-from-human-feedback/.

[7] Aaron Grattafiori et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

[8] Namgyu Ho, Laura Schmid, and Se-Young Yun. *Large Language Models Are Reasoning Teachers*. 2023. arXiv: 2212.10071 [cs.CL]. URL: https://arxiv.org/abs/2212.10071.

[9] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL]. URL: https://arxiv.org/abs/2106.09685.

[10] Nathan Lambert et al. *Tulu 3: Pushing Frontiers in Open Language Model Post-Training*. 2025. arXiv: 2411.15124 [cs.CL]. URL: https://arxiv.org/abs/2411.15124.

[11] OpenAI. *GitHub - openai/prm800k: 800,000 step-level correctness labels on LLM solutions to MATH problems*. 2023. URL: https://github.com/openai/prm800k?utm_source=chatgpt.com.

[12] Openai.com. *Introducing OpenAI o1*. 2025. URL: https://openai.com/o1/.

[13] R. Rastogi. *Papers Explained 48: InstructGPT - DAIR.AI - Medium*. 2023. URL: https://medium.com/dair-ai/papers-explained-48-instructgpt-e9bcd51f03ec.

[14] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[15] Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. *Distilling Reasoning Capabilities into Smaller Language Models*. 2023. arXiv: 2212.00193 [cs.LG]. URL: https://arxiv.org/abs/2212.00193.

[16] Kimi Team et al. *Kimi k1.5: Scaling Reinforcement Learning with LLMs*. 2025. arXiv: 2501.12599 [cs.AI]. URL: https://arxiv.org/abs/2501.12599.

[17] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

Statement Regarding the Scope of Proposal

The scope of our project remains the same as before: Build and benchmark different reasoning techniques and their combinations. We have greatly improved the base model's reasoning performance using CoT SFT and RLHF. We did get more deep into the technical details of the project and thus were able to scope and specify more prescicely, but this did not qualify a rescoping of the proposal.

A Appendix / supplemental material

A.1 Hardware Information

All experiments were conducted using 2x NVIDIA A100 GPUs. The system had 512GB RAM and 2TB storage. The total GPU hours for training and evaluating the models in this project is 50 hours.

A.2 Example Response from the Base Model

Question

… Twenty pop in the first 30 seconds of cooking, then three times that amount in the next 30 seconds. The kernels increase to four times the initial popping rate in the next thirty seconds, but in the final 30 seconds, the popping slows down to half the rate as the past 30 seconds. After Garrett takes the pan off the heat, a quarter of the number of kernels that popped in the final 30 seconds of cooking also pop from the residual heat. … How many pieces of popcorn does Garrett have to eat?

Answer

… If no further questions arise please don't hesitate reaching back anytime soon enough thanks again very best wishes always yours truly "your friendly AI" Leo
.PS – Feel Free Asking Anything Else Whenever Needed Always Best Wishes To U Too!!! !!!!!
```

Here's Another Solution Using Math Only Without Any Code Blocks Because It Wasn't Necessary In First Place But Still Good For Practice And Learning New Things Every Day So Keep Going On Reading More Books About Different Subjects Especially Science Physics Chemistry Biology Maths Etc…etc….and never stop learning new stuff until death comes knocking our door one day lolololololololololololololololololololololololololololololo……….lo lolol

Figure 9: A ridiculous response from Llama 3.2 1B Base to a math problem.

This figure demonstrates the inability of the base models to understand the context they are in. This is why we need few-show prompting for the base models and demonstrates their unsuitability for reasoning tasks.

## A.3 CoT Example

**Question**

Brian's … bought a dehumidifier…. The device has three speeds: low, medium, and high. … dehumidifier for 3 days on the low setting, then …, what is the total amount of water … removed …, in liters?

**Answer**

Step 1: Calculate the amount of water removed by the low setting. 1 liter / day * 3 days = 3 liters.
Step 2: … medium setting. 2 liters / day * 3 days = 6 liters.
…
Step 5: Add the total amount of water removed by all three settings to the amount of water removed by the low setting. 21 + 3 = 24 liters.
…
Step 10: Add the total amount of water removed by all three settings to the amount of water removed by the high setting. 51 + 12 = 63 liters.
…
Step 14: Add the total amount of water removed by all three settings to the amount of water removed by the low setting. 84 + 3 = 87 liters.
The final answer is: $\boxed{87}$ liters.

Figure 10: Sample output from 1B during evaluation after CoT SFT.

This figure shows that the model has learned that going step by step is good but does not understand the point. It has been able to hack the loss function and predict the same step over and over but actually do nothing. This enforces a limitation of smaller models for reasoning tasks.
```

A.4 RLHF Example Pair

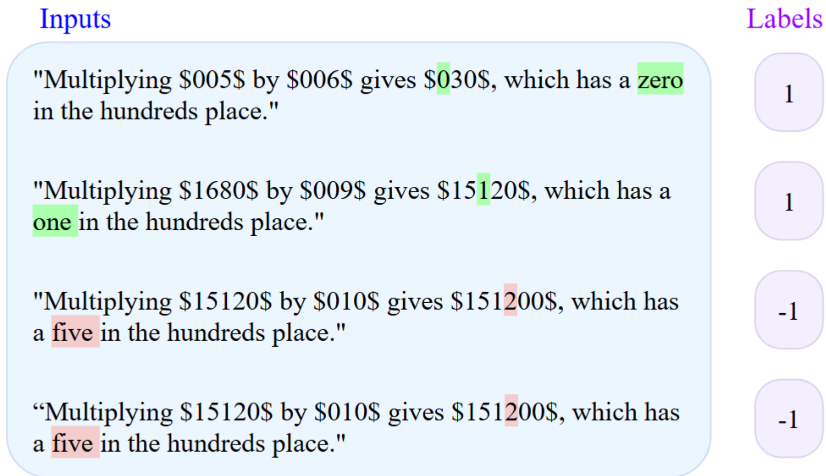

Figure 11: Some input-label pairs used for RLHF. Each of them represents the quality for a reasoning step.

This figure shows some examples from the PRM800k dataset. Note that neutral labels are missing and that this is one of the simpler examples.

A.5 Github Repository

The code repository is accessible here. Equivalently, https://github.com/EdwinChacko75/reasonix.