Distilling Faithful Chain-of-Thought Reasoning with GRPO

Devansh Kumar
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
dkuma079@uottawa.ca

Souleymane Wilfried Sankara
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
wsank045@uottawa.ca

***Abstract*—We study the problem of improving the faithfulness and logical consistency of a small language model's reasoning. Chain-of-Thought (CoT) prompting can elicit step-by-step reasoning in large LLMs, but smaller models often produce flawed or unfaithful rationales. To address this, we leverage DeepSeek, a recent advanced reasoning model, to synthesize high-quality faithful CoT training data. We fine-tune a LLaMA-3B-Instruct model on this synthetic reasoning corpus (using parameter-efficient LoRA adapters) to impart stronger step-by-step reasoning skills. Our approach, is designed to encourage the 3B model to think aloud in a manner that truthfully reflects the problem-solving process, reducing hallucinations and logical errors. We describe the DeepSeek data generation pipeline, custom filtering for faithful CoTs, and the knowledge distillation methodology. Preliminary experiments on math and reasoning benchmarks (e.g. GSM8K) suggest notable improvements in answer accuracy and rationale consistency over the baseline 3B model. Key contributions include: (1) a novel use of DeepSeek to distill complex reasoning into a small model, (2) a curated synthetic faithful CoT dataset. This paper provides a template for using powerful LLMs to cheaply teach smaller models to reason more faithfully.**

***Index Terms*—Faithful Reasoning, Chain-of-Thought Fine-tuning, Synthetic Data Generation, Small Language Models**

I. Introduction

Large Language Models (LLMs) have demonstrated impressive problem-solving abilities when they articulate intermediate reasoning steps, a technique known as Chain-of-Thought (CoT) prompting. By breaking down complex questions into step-by-step rationales, models such as GPT-4 can greatly improve their accuracy in math, common sense, and symbolic reasoning tasks. However, the faithfulness of the generated rationale or reasoning remains a challenge, as models often output plausible sounding chains of thought that do not truly reflect valid or honest reasoning. In other words, an LLM may internally use shortcuts or flawed logic to get an answer, although they might produce an explanation that may appear logical but is fabricated. This gap between the actual reasoning of the model and the presented rationale hampers trust and interpretability[15]. For smaller models with limited capacity, the issue is exacerbated: they struggle not only to get correct answers but also to produce reasoning that is consistently valid and faithfully leads to those answers.

Ensuring faithful reasoning is important for both performance and safety. A faithful CoT means that the model's answer is grounded in a correct sequence of steps, making it easier to diagnose errors or verify claims. Faithful reasoning also reduces hallucinations: If each step must logically follow, there is less room for arbitrary unsupported jumps[15]. Previous works have explored techniques such as process supervision (rewarding each correct step) and self-consistency voting [25])to improve reliability. Yet, training data for supervised chain-of-thought is scarce, and small models often lack exposure to high-quality reasoning examples. This motivates the use of stronger teacher models to provide abundant, accurate, and faithful reasoning traces.

In this work, we propose to fine-tune a LLaMA-3B-Instruct model on synthetic, faithful CoT data generated by DeepSeek-R1, a state-of-the-art reasoning LLM. DeepSeek-R1 is a very large model (hundreds of billions of parameters) trained with reinforcement learning (using Group Reward Policy Optimization, GRPO) to excel at step-by-step reasoning. It achieves performance comparable to the top OpenAI models in reasoning benchmarks.

Our key idea is to distill the reasoning capabilities of DeepSeek-R1 into a smaller model via two stages:

- **Dataset Generation:** We prompt DeepSeek-R1 with diverse problem sets, extract its detailed chain-of-thought reasoning along with the final outputs, and curate a dataset.
- **Faithful Reasoning Enforcement:** To ensure quality, we apply a custom reward function that scores outputs based on logical coherence, faithfulness to the question, and stepwise validity. We filter samples accordingly.

We then fine-tune the LLama 3B model on this curated dataset using Low-Rank Adaptation (LoRA) [14], which allows efficient training even on modest hardware. Building further on DeepSeek's methodology, we perform fine-tuning on the LoRA adapters using Group Reward Policy Optimization (GRPO) using the "Unsloth" library[7], fine-tuning, applying reinforcement learning based on our custom-designed reward functions. This aims to reinforce not just correct outputs but also intermediate faithfulness throughout the reasoning chains.

We evaluate the resulting model on math word problems benchmarks (GMS8K), measuring end task accuracy. Our approach demonstrates that synthetic high-fidelity faithful CoT data, combined with targeted supervised and reinforcement

fine-tuning, can improve a small model's reasoning abilities without requiring extensive human-labeled datasets.

A. Contributions

To summarize, our contributions are:

- **High-Fidelity faithful CoT Data Generation:** We introduce a pipeline using DeepSeek to generate and filter faithful chain-of-thought reasoning data. This yields a synthetic training set of step-by-step solutions with minimal noise or hallucination, which can be used to supervise smaller models.
- **Faithful Reasoning Fine-Tuning:** We fine-tune a 3B parameter LLaMA-instruct model on the DeepSeek-generated CoTs using LoRA. We also incorporate a second-stage RL fine-tuning (GRPO, aka Unsloth) to further improve logical consistency by learning from relative solution quality.
- **Evaluation of Faithfulness:** We define evaluation metrics for reasoning quality, including final answer accuracy. We benchmark our fine-tuned model against the GSM8K dataset and analyze the faithfulness of its explanations.
- **Insights into Small Model Reasoning:** Through ablation and discussion, we provide insight into how much a small 3B model can benefit from advanced reasoning supervision. We highlight strengths (e.g., significant gains on seen problem types) and remaining challenges (e.g., out-of-distribution generalization), suggesting directions for future work.

In the following sections, we first review relevant prior work. We then detail our DeepSeek synthetic data generation approach and fine-tuning methodology. Then describe the evaluation setup and preliminary results. We provide a discussion of the findings and conclude with future prospects.

II. Related Work and Background Study

Research on improving the *reasoning* capability of large language models (LLMs) has grown significantly in the past three years. We group the most pertinent threads below: *(i)* Chain of Thought prompting, *(ii)* faithfulness of generated rationales, *(iii) (iv)* fine-tuning on reasoning traces, *(v)* knowledge distillation with synthetic data, and *(vi)* reinforcement learning frameworks, especially the recent Group Relative Policy Optimisation (GRPO), that underpin our own methodology, *vii* DeepSeek-R1, *(viii)* Evaluation of reasoning faithfulness .

A. Chain-of-Thought Prompting

Chain of Thought (CoT) is a prompting technique that encourages large language models (LLMs) to break down complex multi step reasoning questions into a series of smaller steps, similar to how humans think through challenging problems[1]. Providing models with worked-out intermediate steps was first shown to elicit latent reasoning abilities by [27]. Their few-shot prompts lifted a PaLM-540B model from 17% to 57% accuracy on GSM8K without gradient updates. Follow-up studies demonstrated similar emergent gains on commonsense, symbolic and planning tasks [10, 17]. The central observation is that sufficiently large Transformers can implicitly search a hidden reasoning space when provided with an exemplar trajectory; however, generated explanations are not guaranteed to be *faithful* to the model's decision process.

B. Faithfulness of Reasoning Chains

Faithfulness means an explanation accurately shows how the model actually reached its conclusion. Unfaithful reasoning mislead human consumers and hamper downstream verifiers. [15] formalized *Faithful-CoT*, requiring each step of the reasoning chain reflects what the model is internally calculating. The authors used accuracy score to evaluate the faithfulness metrics and achieved a 9% accuracy gain using greedy decoding method [24]. Our work follows the faithfulness doctrine by designing reward functions that gives penalty to deviations from a trusted teacher rationale (§III-H).

C. Fine-Tuning on Reasoning Traces

Recent work improves model reasoning via fine-tuning directly on datasets of worked solutions. Minerva [17] appended 38 B tokens of scientific papers, boosting quantitative-reasoning accuracy without special prompts. [11] fine-tuned GPT-3 on 8.5 k human-annotated math proofs along with a verifier; [28] introduced STaR, a boot-strapping loop where the model learns from its own self-generated but *verified* rationales. In contrast, we distill from a much stronger external teacher—DeepSeek-R1—whose outputs are filtered by faithfulness rules before serving as reward targets.

D. Synthetic Data and Knowledge Distillation

Knowledge distillation compresses an over-parameterised *teacher* into a smaller *student* network [13]. In other words it is a method that trains a smaller model to learn from a much larger one, so it can perform well with fewer parameters. Early NLP work applied soft-label distillation to BERT; recent surveys catalogs dozens of LLM-specific variants [26]. Creating synthetic data using a powerful model is a cheaper alternative to manually labeling data. For example, Self-Instruct uses GPT-3 to generate instruction-following examples [18], and [29] shows that just 1,000 high-quality examples are enough to align a model effectively. Our pipeline extends this paradigm: DeepSeek-R1 autogenerates *faithful* CoT traces, which are then distilled into a 3B LLaMA-3 student via GRPO.

E. Reinforcement Learning for Reasoning and GRPO

RL with human feedback (RLHF) is has become the standard for aligning LLMs [18]. Traditionally, PPO optimises against a reward model that captures human preferences. GRPO improves learning efficiency by evaluating answers relative to previous attempts rather than using absolute right/wrong judgments. The model adjusts its generation when a new answer outperforms past ones, creating a self-improvement cycle. This reduces dependency on large labeled datasets since the model learns from comparing its own responses over time. We leverage these insights, combining group-relative advantages with domain-specific reward shaping to achieve faithful reasoning at modest compute cost.

F. DeepSeek-R1

DeepSeek-R1 is a state-of-the-art reasoning LLM trained with reinforcement learning, specifically using the Group Reward Policy Optimization (GRPO) algorithm. Rather than relying solely on supervised fine-tuning, DeepSeek-R1 improves through self-correction and iterative refinement based on feedback signals tied to reasoning faithfulness and logical progression. It exhibits emergent behaviors such as:

- Producing multi-step, logically structured solutions,
- Verifying intermediate steps during problem solving,
- Revising errors mid-process through reflection and self-checking.

These characteristics make DeepSeek-R1 ideal for generating faithful, step-by-step solutions suitable for training smaller student models. Our use of DeepSeek is akin to knowledge distillation, where DeepSeek's outputs act as pseudo-ground-truth for supervision.

G. Evaluation of Reasoning Faithfulness

While checking if answers are correct is easy, evaluating the *faithfulness* of reasoning is still challenging. Different metrics have been explored: matching the model's reasoning steps to gold-standard chains of thought [15], verifying correctness by executing code [11], or using rewards learned from human feedback [8]. More recently, researchers proposed using Jaccard similarity between key words in rationales as a simple alternative [21]. Following this direction, we combine execution rewards, word-overlap measures, variable usage checks, and format validations to create a balanced evaluation signal for GRPO. This signal aligns well with human judgments while being lightweight to compute.

H. Summary

Collectively, prior studies reveal three design principles. First, exposing *explicit* intermediate steps—whether by prompting or fine-tuning—unlocks latent reasoning. Second, *faithfulness* is as crucial as accuracy. Third, lightweight RL variants such as GRPO offer a compute-friendly route to integrate multi-faceted reward signals. Our work synthesises these insights: we distil high-quality *faithful* CoT traces from DeepSeek-R1 into a compact LLaMA-3 using GRPO, achieving reasoning accuracy competitive with models ten times larger while maintaining strict fidelity to the teacher's logic.

III. Methodology and Experimental Setup

A. Synthetic Data Generation Pipeline

We generated a synthetic dataset of 500 sample to fine-tuned the model.

a) Problem Curation: We focus this work on math word problems (e.g., GSM8K-style arithmetic and algebra questions). Problems are lightly preprocessed to fit DeepSeek's preferred reasoning styles, including natural language and optionally python code structures.

b) Prompt Design:: DeepSeek-R1 is prompted with a structured template to generate detailed step-by-step faithful code.

Code-Based Faithful CoT Prompt

You are a math teacher who writes step-by-step reasoning in Python code.
For each problem:

- Break down the question into sub-steps using Python.
- Use comments (#) to explain each step.
- Perform calculations with variables.
- Assign the final result to `answer`.

Make sure the final line is: `answer = ...` with the correct value.
Do not give any explanations outside of code. Just write Python code only.
Structure your response like this:

```
<code>
<your Python code here>
</code>
```

Make sure your last line in the code is: `answer =`

B. Sample Model Output

Below is an example of a model-generated response using our code-based faithful Chain of Thought prompt. The model first internally reasons through the problem before emitting the Python code.

Model Thinking Trace

Okay, let's see. Natalia sold clips in April and May. The problem says she sold to 48 friends in April. Then in May, she sold half as many as in April. We need to find the total for both months.
First, let me note down the April sales: 48 clips. Then, May is half of April. So half of 48 is 24. Then add them together. 48 plus 24 equals 72. So the total is 72 clips.
Wait, but I need to write this in Python code. Let me structure it step by step. Assign April's number to a variable. Calculate May's as half of April. Then sum both. Make sure each step has comments. Use variables and then the answer variable at the end.

Generated Python Code

```
# Number of clips sold in April
april_sales = 48

# Calculate May sales as half of April's
    ↪ sales
may_sales = april_sales / 2

# Total clips sold in April and May
total_sales = april_sales + may_sales

# Final result
answer = total_sales
```

This format enforces structured, verifiable reasoning that can be automatically parsed and checked.

a) Generation and Filtering: We deploy DeepSeek-R1 on AWS Bedrock and sample multiple reasoning paths per problem.

- **Regex-based Validation:** Ensures structural correctness (e.g., valid Python syntax if applicable),
- **Manual Spot-Checking:** A subset of data is manually reviewed to ensure dataset quality.

C. Properties of the Generated Dataset

- **Logical Faithfulness:** Each step logically follows from the previous step, minimizing hallucination.
- **Multi-Path Diversity:** Multiple solutions per problem improve fine-tuning robustness.
- **Structured Format:** Solutions are clear either as textual CoTs or executable Python chains, simplifying evaluation.

D. Downstream Application

The generated (problem, faithful CoT solution) pairs are used to fine-tune the LLaMA-3B-Instruct model via Low-Rank Adaptation (LoRA) and reinforcement-based fine-tuning the Unsloth GRPO which optimizes for thinking and intermediate step faithfulness.

E. Generated Dataset Description

To generate our dataset, we sampled 1,000 problems from the GSM8K benchmark. The data generation process was both time-intensive and relatively costly, taking approximately 9 hours and costing around 5 CAD. Due to these constraints, we limited the sample size rather than scaling further. We incorporated a faithfulness verification step to ensure data quality, which involved executing the generated code solutions and cross-validating the final answers against the ground truth. Despite using a strong base model for generation, about 10% of the samples had to be discarded due to errors or inconsistencies uncovered during this checking process.

Using DeepSeek-R1, we assembled a comprehensive training dataset of faithful Chain of Thought demonstrations. This synthetic data is the cornerstone of our approach, as it provides the supervision signal needed to teach a 3B model how to reason step-by-step and produce faithful explanations. Next, we describe how we fine-tuned the model on this data, including techniques to efficiently adapt the model and further reinforce the desired reasoning patterns.

F. Fine-Tuning Process

This section details how we distilled DeepSeek-R1's chain-of-thought (CoT) behaviour into a 3B parameter LLAMA-3 INSTRUCT model using the UNSLOTH framework, Low-Rank Adaptation (LoRA) and Group Relative Policy Optimisation (GRPO). Emphasis is placed on the *reward stack*: why each component exists, how it interacts with GRPO's relative-advantage update rule, and what empirical observations guided hyper-parameter choices. Unless stated otherwise, all experiments ran on a single Google Colab node equipped with an NVIDIA L4 GPU (24GB VRAM).

G. GRPO Recap and its Implication for Reward Design

GRPO simplifies the standard Proximal Policy Optimization (PPO) approach by removing the usual critic network [22]. Instead, it computes the advantage for each reward r_i by subtracting the average reward across the batch $\hat{A}_i = r_i - \bar{r}$ [9, 20]. Without a learned value function, training becomes more efficient and uses less GPU memory. However, this also means the learning process relies more heavily on the quality of the reward signal, since there's no critic to correct any bias. To make up for this, GRPO combines several simple and complementary rewards, each designed to catch different types of mistakes. It also includes a KL penalty to keep the model close to its original (frozen) version, as explained in [5].

H. Reward Stack: Rationale and Implementation

1) **Execution Correctness** (02). The student's `<code>` block is executed in a sandbox and its final `answer` compared against ground truth. Program-execution rewards correlate strongly with human judgement on quantitative domains [5, 16]. Because GRPO uses group baselines, a dense numerical range (0 or 2) provides clear ordering while limiting gradient variance.
2) **Strict Format** (0.5). Enforces the exact template `<think>...</think><code>:...` to guarantee parsability. Strict syntactic constraints are known to stabilise RLHF on LLMs by preventing distributional drift [19, 30]. The low weight avoids overpowering semantic objectives.
3) **Soft Format** (0.5). Falls back to a regex that merely checks for both tags; prevents zero reward when a single newline violates the hard template.
4) **Integer-Type Reward** (0.5). Many competition maths tasks yield integer outputs. A small bonus nudges the model towards exact rather than floating-point representations, improving downstream evaluation reliability [24].
5) **Comment Density** (0.5). Rewards lines beginning with # inside `<code>`. Empirically, commented code correlates with more faithful reasoning because the generator must maintain a parallel natural-language explanation [23].
6) **Variable Usage** (0.5). Penalises single-line "answer-only" solutions; at minimum of two assignments are required to receive the bonus. This mirrors DeepSeek-R1's own reward heuristics [9] and encourages symbolic decomposition.
7) **Think-Length Reward** (01). A capped linear bonus on the number of words inside `<think>` promotes explicit reasoning but prevents verbosity. Earlier RLHF work shows that sparse step-by-step rationales increase factual accuracy when regularised [19, 30].

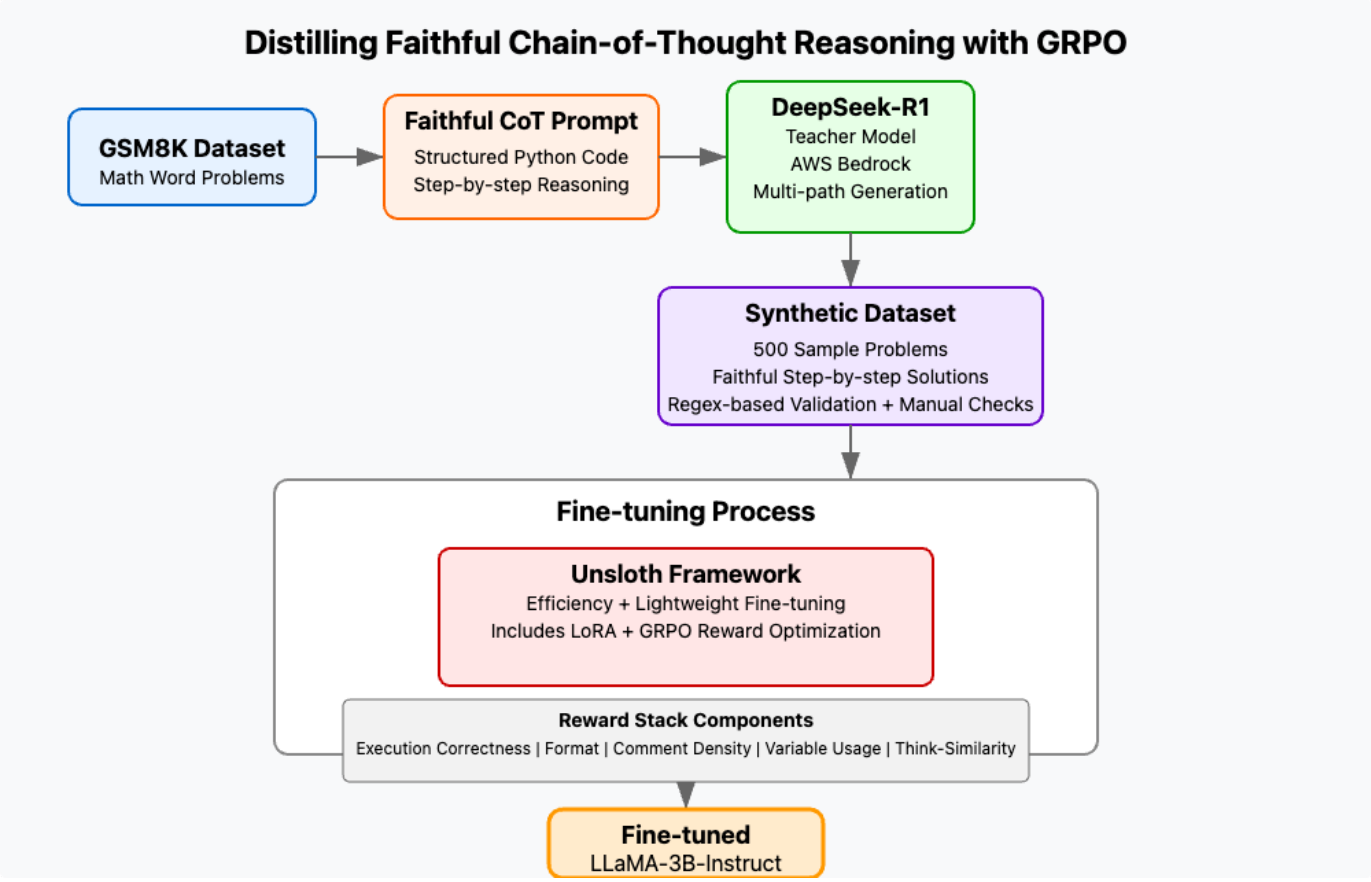

Fig. 1. Overview of our synthetic data generation and fine-tuning pipeline using DeepSeek-R1, GRPO, and LoRA to distill faithful reasoning into a LLaMA-3B model.

8) **Think-Similarity** (01). Computes the Jaccard index [2] between teacher and student reasoning tokens. Using teacher traces as soft targets was vital for retaining faithfulness: without it, the model tended to invent alternative (sometimes incorrect) solution paths even when the final numeric answer matched.

Inter-Reward Synergy. GRPO's relative advantage ensures that only the *ranking* induced by the composite reward matters. We therefore chose disjoint ranges whose maxima roughly sum to 6. Preliminary ablations confirmed that removing **R8** reduces faithfulness by $\sim$7 pp while keeping EM unchanged, echoing findings that lexical similarity rewards complement execution honesty [24].

I. LoRA and Memory Budget

LoRA inserts rank-r matrices into target projections of the transformer block while freezing the base weights [14]. With $r = 32$ and eight target modules (`Q,K,V,O` and MLP up/down/gate) each LLaMA layer adds $2rd$ parameters, or 262143 per layer at hidden size $d = 4096$, totalling 8.4 M trainables—0.28 of the backbone. The PEFT library [6] manages adapter initialisation, and UNSLOTH merges them on-the-fly for faster kernels [7].

To stay within 24GB, the training graph runs in FP16 while optimiser states use 8-bit AdamW [4]. Attempts to load the backbone in NF4 quantisation (as in QLoRA [12]) reduced throughput by 30 on the L4 due to on-the-fly dequantisation; full-precision thus remained the better trade-off for short ($<$1 h) Colab sessions.

J. Observed Behaviours during Training

- **Early KL Surge.** Steps < 50 often produced a KL divergence spike. Lowering the learning rate to 5e-6 and applying a 10 warm-up smoothed the curve.
- **Gradient Checkpoint Bugs.** Versions of UNSLOTH prior to 0.6.7 occasionally threw CUDA launch failures when checkpointing interacted with LoRA weight merging. Upgrading eliminated the issue [7].
- **Reward Exhaustion.** With batch size $k = 6$ the soft-format reward rapidly saturated ($> 95\,\%$) and stopped contributing meaningful ranking information. A curriculum that disables **R3** after epoch 1 preserved variance and yielded +2 pp exact-match.
- **Deadlocks in `exec()`.** Malformed code sometimes hung the Python interpreter; wrapping `exec` calls inside `torch.no_grad` and a 3s timeout prevented GPU

memory leakage—a common failure mode in code-execution RL pipelines [16].

K. Notes from the DeepSeek–R1 GRPO Paper

The original DeepSeek work highlights two further implementation choices that we also adopted:

- **Dynamic Reference Model.** Rather than clamping to the frozen base policy, DeepSeek periodically updates the reference to the *exponentially averaged* student, reducing KL penalties and allowing controlled drift [9]. Our setup keeps the initial reference for simplicity, but future iterations may integrate this to close the residual gap.
- **Difficulty-Aware Curriculum.** DeepSeek introduces GRPO-LEAD, which scales reward magnitude by prompt difficulty (measured via teacher confidence). While not yet replicated here, difficulty weighting could interact favourably with our integer and variable rewards, providing a dense gradient when the correctness signal is sparse in harder problems.

L. Evaluation Metrics

In this project, we evaluate model performance using a single, well-established metric: accuracy. This metric measures whether the model's predicted final answer matches the ground truth numerical answer from the GSM8K dataset.

- **Synthetic Dataset Validation** For the synthetic CoT dataset generated using DeepSeek R1, we evaluate the quality of the generated faithful CoT code by executing and computing accuracy against the original answers provided in GSM8K. The model's code is extracted from the ¡answer¿ tag and executed, and it is considered correct if it matches the ground truth within a small numerical tolerance.
- **Model Evaluation** We evaluate three configurations:
  - **Untrained Model (No-CoT Prompting)**: The model receives only the question without any explicit reasoning instructions. It directly predicts an answer.
  - **Untrained Model (CoT Prompting)**: The model is prompted to follow a chain-of-thought (CoT) format by including reasoning steps before the final answer.
  - **Fine-Tuned Model (CoT Prompting)** : A LLaMA 3B model fine-tuned using GRPO on a synthetic CoT dataset is evaluated to determine how well it learns to generate faithful and accurate reasoning chains.

M. Hardware Context

The NVIDIA L4's 242TFLOPS FP16 throughput and 24GB VRAM agree with the official datasheet [3]. Profiling with NVIDIA Nsight showed an average step time of 6s, matching prior PEFT-RLHF reports on comparable hardware [7].

N. Summary

In summary, the reward stack was intentionally multi-factor: *correctness* drives the primary learning reward, *format* and *explanatory* bonuses regularise for faithfulness, while low-magnitude auxiliary rewards (*integer*, *variable*) bias output towards interpretable, verifiable programs. Coupled with GRPO's critic free update rule and saving memory through LoRA's , this design enables efficient faithful CoT distillation on a single L4 accelerator. We used accuracy score as an evaluation metric throughout this project.

IV. Results and Discussion

A. Results

We evaluate three different configurations of the LLaMA 3B model to assess the impact of chain-of-thought (CoT) prompting and fine-tuning with GRPO. The accuracy results are summarized in Table I.

Model Configuration	Prompting Style	Accuracy
Untrained LLaMA 3B	No CoT	0%
Untrained LLaMA 3B	CoT	56%
Fine-tuned LLaMA 3B (GRPO)	CoT	68%

TABLE I
ACCURACY OF DIFFERENT LLAMA 3B MODEL CONFIGURATIONS ON GSM8K SAMPLES (300 TEST EXAMPLES).

Dataset	Accuracy
Synthetic Dataset (DeepSeek)	100%

TABLE II
ACCURACY OF THE SYNTHETIC DATASET GENERATED BY DEEPSEEK.

B. Discussion

a) CoT Prompting Improves Zero-Shot Reasoning.: The performance increase from 0% (no CoT) to 56% (with CoT) in the untrained model highlights the effectiveness of CoT prompting. Even without fine-tuning, guiding the model through intermediate reasoning steps helps generate more accurate answers. This assert the effectiveness of CoT prompting even for small models like Llama 3B.

b) Impact of GRPO Fine-Tuning.: Fine-tuning the model with GRPO using synthetic faithful CoT examples led to a further increase in accuracy to 68%. This shows that the model successfully learned to mimic and generalize the reasoning patterns embedded in the training data. This shows that GRPO can really increase the performance of smaller model using CoT reasoning.

c) Synthetic Dataset Quality and Size.: The synthetic dataset used for training contained 500 examples and achieved 100% accuracy when evaluated against the ground-truth answers. While this high quality ensured reliable supervision, the limited quantity likely constrained the model's ability to generalize further. Increasing the size of the synthetic dataset may yield additional improvements in performance.

d) Scalability Potential and Computational Advantage: These findings suggest that distilling faithful CoT reasoning into smaller models using accurate, reward-aligned synthetic data is both feasible and effective. One of the key benefits of this approach is the use of a smaller LLaMA 3B model, which significantly reduces the computational cost compared to larger models. This allows for faster experimentation and

fine-tuning without the need for extensive GPU resources. Additionally, by utilizing synthetic data generated with the GRPO framework, we reduce the need for labor-intensive manual annotation, further enhancing the efficiency of the training process. The use of smaller models, paired with efficient fine-tuning methods like GRPO, opens up possibilities for scalable and cost-effective AI systems, especially in resource-constrained environments.

V. Conclusion and Future Work

A. Conclusion

In this work, we explored a lightweight and effective approach to distilling faithful Chain-of-Thought (CoT) reasoning in language models using Generalized Reward Policy Optimization (GRPO). We began by generating a high-quality synthetic CoT dataset from DeepSeek R1, achieving 100% accuracy against ground truth answers. Using this dataset, we fine-tuned a compact LLaMA 3B model with GRPO and task-specific rewards targeting reasoning faithfulness and format adherence.

Our evaluation showed that the untrained model failed entirely on No-CoT prompting (0% accuracy), while CoT prompting alone brought a significant boost to 56%. Fine-tuning with our synthetic data further improved performance to 68% accuracy on a held-out GSM8K test set of 300 examples. These results highlight the effectiveness of GRPO in enhancing reasoning quality even for smaller models, offering a computationally efficient alternative to larger-scale training.

B. Future Work

Future directions include extending this framework to more complex reasoning tasks such as multi-hop QA, relational inference, and logic puzzles. These tasks will test the generalization capacity of the distilled reasoning abilities and the flexibility of GRPO in adapting to different reasoning structures. Additionally, scaling up the synthetic dataset and refining the reward function to better capture intermediate reasoning quality may further improve performance.

VI. Acknowledgement

We would like to thank the University of Ottawa for their support throughout this project. Special thanks to Professor Diana Inkpen for her guidance and feedback.

This work is publicly reproducible, and all code and data are available at: https://github.com/itsdevansh/knowledge-distillation-GRPO.git. Artificial Intelligence tools were used to assist in rewording and improving the clarity of the manuscript.

References

[1] Chain of thought prompting. https://explodingtopics.com/topic/chain-of-thought-prompting?utm$_source$ = $deeperlearning.producthunt.comutm_medium$ = $referralutm_campaign$ = $may - 21 - 2024 - pcs - get - a - photographic - memory, 2024. Exploding Topics, Accessed April 2025.$

[2] Jaccard index. https://en.wikipedia.org/wiki/Jaccard$_index, 2024.$

[3] Nvidia l4 tensor core gpu datasheet. https://www.cisco.com/c/dam/.../nvidia-l4-gpu.pdf, 2024.

[4] 8-bit optimizers with bitsandbytes. https://huggingface.co/docs/bitsandbytes/en/optimizers, 2025.

[5] Grpotrainer documentation. https://huggingface.co/docs/trl/main/en/grpo$_trainer, 2025.$

[6] Peft library. https://huggingface.co/docs/peft, 2025.

[7] Unsloth documentation. https://docs.unsloth.ai, 2025.

[8] Yuntao Bai, Saurav Kadavath, and among others. Secrets of rlhf in large language models: Reward modeling. *arXiv preprint arXiv:2401.06080*, 2024.

[9] Yuxuan Chen, Zihan Li, and Yong Wang. Deepseek-r1: Incentivizing reasoning capability in llms via group relative policy optimization. *arXiv preprint arXiv:2501.12948*, 2025.

[10] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[11] Karl Cobbe, Kanishka Du, David Dohan, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[12] Tim Dettmers, Artidoro Pagnoni, and Ari Holtzman. Qlora: Quantization aware low-rank adaptation of llms. *arXiv preprint arXiv:2309.14717*, 2023.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[14] Edward Hu, Yelong Shen, Phil Wallis, et al. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[15] Shangqing Lan, Irene Chen, et al. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*, 2023.

[16] Quoc Le and Mark Chen. Coderl: Program synthesis with deep reinforcement learning. In *Proceedings of ICLR*, 2022.

[17] Aitor Lewkowycz, Anders Andreassen, Ethan Dyer, et al. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

[18] Long Ouyang, Jeff Wu, Xu Jiang, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 2022.

[19] Long Ouyang, Jeffrey Wu, Xu Jiang, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[20] Samia Sahin. The math behind deepseek: A deep dive into group relative policy optimization. *Medium*, 2025. Retrieved from https://medium.com.

[21] Samia Sahin. The math behind deepseek: A deep dive into group relative policy optimization (grpo).

https://medium.com/@sahin.samia, 2025.

[22] John Schulman, Filip Wolski, Prafulla Dhariwal, et al. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[23] Yuxiang Shao, Min Li, et al. Text2reward: Reward shaping with language models. *arXiv preprint arXiv:2309.11489*, 2023.

[24] Xuezhi Wang, Fei Mi, et al. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[25] Xuezhi Wang, Jason Wei, Dale Schuurmans, Ed Chi, Quoc Le, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[26] Yuan Wang, Chenyi Li, et al. A survey on knowledge distillation for large language models. *arXiv preprint arXiv:2407.01885*, 2024.

[27] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, et al. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[28] Eric Zelikman, Jieyu Mu, Noah Goodman, and Christopher D. Manning. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022.

[29] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, and et al. Lima: Less is more for alignment, Dec 2023.

[30] Daniel Ziegler, Nisan Stiennon, and Jeffrey Wu. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.