Training Reasoning Sub-Skills in LLMs with Synthetic Data

Cormac Cureton
M.Sc. Student
Electrical and Computer Engineering
McGill

Abhijeet Praveen
M.Sc. Student
Electrical and Computer Engineering
McGill

Xiaoyin Chen
Phd Sudent, Project Mentor
Computer Science
Université de Montreal

Abstract

This work investigates the potential of fine-tuning large language models (LLMs) using programmatically generated synthetic data to enhance their reasoning sub-skills. The study focuses on search as a foundational reasoning sub-skill and evaluates its transferability to higher-order reasoning tasks, specifically Sudoku and Zebra puzzles. Using Low-Rank Adaptation (LoRA), we fine-tune LLMs on synthetic search trajectories without increasing inference-time computational costs, addressing the challenge of high-quality reasoning data scarcity. Our experiments employ Partial Accuracy and Strict Accuracy metrics to assess the effectiveness of fine-tuning and highlight task-specific performance variations. Results demonstrate that fine-tuning on synthetic search trajectories offers marginal improvements in zero shot Zebra puzzle performance compared to the base model. Synthetically fine-tuned models don't offer any improvements when tested on Sudoku. As expected, models fine-tuned directly on task-specific datasets consistently outperform search-fine-tuned models, emphasizing the value of task-specific data. This study underscored the difficulty in improving model attributes which generalize across tasks, We release all our code to facilitate future research into scalable, efficient methods for enhancing LLM reasoning capabilities. [1]

1 Introduction

In recent years, large-scale language models have grown hugely in popularity, being successfully applied to a wide-range of tasks. One factor which seems to be holding back language models from larger use is their struggle to reason effectively, despite huge knowledge bases, LMs still struggle with reasoning tasks (Valmeekam et al., 2022; Stechly et al., 2024). Approaches like Chain of Thought (Wei et al., 2023), self-consistency (Wang et al., 2023), and step by step verifiers (Lightman et al., 2023) have all been explored to improve performance on reasoning tasks. These approaches improve performance but they also heavily increase the computational costs at inference time.

Instead, our work looks to fine-tune language models for improved performance on reasoning tasks with a single inference pass. By using Low-Rank Adaptation (LoRA), transformer-based language models can be adapted without increasing computational costs at inference-time (Hu et al., 2021).

However, a challenge with improving reasoning in language models is that high-quality reasoning traces are difficult to find and expensive to produce (Bansal et al., 2024). The lack of high-quality reasoning data motivates the desire to fine-tune models with synthetic data. Unfortunately, generating reasoning traces for training is a non-trivial task and using LMs to create traces comes with significant computational cost (Bansal et al., 2024).

A more computationally efficient method to create synthetic data is programmatic generation. This approach has been has been explored to improve LM performance on skills like search and logic (Gandhi et al., 2024; Lehnert et al., 2024; Pi et al., 2022). We hypothesize that these lower-level skills are necessary for performance on higher-level reasoning tasks. Therefore this work will look to fine-tune a base model on synthetic data to improve a lower-level skill, search. We will then assess whether the gains from fine-tuning on synthetic data will transfer to higher-level reasoning tasks. We select Zebra puzzles and Sudoku as two reasoning tasks for this work, more details about the datasets and evaluation can be found in section 4. This work seeks to provide a path to improve

[1] https://github.com/Cormac-C/llm-reasoning-decomp

model's reasoning ability through fine-tuning on synthetic data for lower-level skills which can be generated at relatively low cost.

Our main contributions are:

1. We open source code which can be used to fine-tune LoRA adapters on reasoning sub-skills with programmatically generated data and measure performance on downstream tasks.

2. We measure the effect that fine-tuning on search trajectories has on performance on higher-level reasoning tasks.

2 Related Work

2.1 Pre-training for Reasoning in Language Models

Pre-training strategies for enhancing reasoning capabilities in language models have been extensively explored, leveraging various auxiliary tasks and external knowledge sources. Notably, ReasonBERT (Deng et al., 2021) integrates knowledge graphs into the pre-training process, enabling the model to establish relationships between entities and improve inference. Similarly, LinkBERT (Yasunaga et al., 2022) incorporates hyperlink structures to uncover semantic relationships, demonstrating improved reasoning in tasks requiring the synthesis of linked information. These methods illustrate how external data sources can augment reasoning but are often computationally expensive and dependent on specific pre-training corpora.

MERit (Jiao et al., 2022) advances reasoning through multi-modal self-supervised learning, combining textual and programmatic reasoning chains. While effective, this approach requires vast datasets and heavy computational resources, limiting scalability. Such reliance on large-scale pre-training highlights a gap in methods designed for specialized reasoning tasks where high-quality data is scarce.

Our work diverges from these approaches by focusing on fine-tuning rather than pre-training. This shift avoids the resource-intensive nature of pre-training while targeting reasoning sub-skills foundational to broader reasoning tasks. By fine-tuning on programmatically generated data, we embed these sub-skills efficiently, enabling transfer to higher-level reasoning challenges.

2.2 Fine-tuning Models on Synthetic Data

Fine-tuning language models using synthetic data has gained prominence for its adaptability to task-specific requirements. Singh et al. (2024) demonstrated the utility of model-generated synthetic datasets for enhancing problem-solving capabilities, achieving state-of-the-art performance on reasoning benchmarks. Similarly, Bansal et al. (2024) utilized smaller language models to generate fine-tuning data for larger models, reducing computational costs while retaining performance. This strategy exemplifies the potential of synthetic data to mitigate resource constraints.

The V-Star framework (Hosseini et al., 2024) employs a model-based verifier trained on solutions from other models, enabling error correction and improved reasoning. Yu et al. (2024) extend this idea by distilling the complex reasoning of "System 2" models into simpler "System 1" models, allowing efficient fine-tuning for tasks like question answering. Furthermore, Ye et al. (2024) augment fine-tuning with both correct and incorrect solutions from the GSM8K dataset, improving error detection and reasoning accuracy.

A distinct line of work explores training on search trajectories. Gandhi et al. (2024) fine-tune models using programmatically generated search logs for text-based games, emphasizing the development of search and logic sub-skills. Similarly, Lehnert et al. (2024) leverage A* search trajectories for planning tasks. While these approaches showcase the effectiveness of trajectory-based data for search and planning, our work expands their scope as we emphasize the transferability of these sub-skills to higher-level reasoning tasks.

2.3 Programmatic Data Generation for Reasoning Tasks

Programmatic data generation has emerged as a cost-effective solution for creating high-quality datasets tailored to reasoning tasks. This approach mitigates the challenges of manual annotation, including time and expense, by algorithmically generating reasoning traces.

Pi et al. (2022) introduced a method for generating reasoning tasks based on program execution, bridging logical reasoning and programming paradigms. Shah et al. (2024) demonstrated that causal transformers trained on programmatically generated puzzles exhibit enhanced logical reasoning, further validating the utility of synthetic data for structured reasoning tasks.

Recent work by Gandhi et al. (2024) and Lehnert et al. (2024) applied programmatic generation to search-based tasks, producing datasets tailored to improve model performance on planning and search problems. While these studies focus on specific sub-skills, our approach extends this paradigm by targeting a broader spectrum of reasoning abilities. By generating synthetic data for sub-skills such as search, logic, and arithmetic reasoning, we enable fine-tuning that is both computationally efficient and effective across diverse reasoning tasks.

2.4 Comparison to Prior Work

Compared to existing works in synthetic data generation, particularly for mathematical reasoning, our approach introduces several distinct advantages. Previous methods, such as those by Ye et al. (2024) and Pi et al. (2022), test their models on a single domain like arithmetic or program execution. In contrast, our work attempts to generalize application of the search sub-skill, testing on different types of reasoning puzzles.

Moreover, unlike Bansal et al. (2024), who rely on language models to generate reasoning traces, we adopt a programmatic generation strategy, a more computationally efficient approach. This distinction opens the door for cheaper scaling, using simpler datasets which can be generated at a fraction of the computational cost.

Finally, Gandhi et al. (2024) showed improvements on search tasks from training on search trajectories. In our work we look to measure whether those gains can generalize to broader reasoning problems, namely Sudoku and Zebra puzzles.

Our work represents a step forward in leveraging programmatic data generation to enhance reasoning capabilities in language models. We look to provide a scalable approach to model fine-tuning for improved performance on diverse reasoning tasks.

3 Models and Approach

This work looks to test whether fine-tuning on search trajectories will show positive transfer to the selected reasoning tasks, Sudoku and Zebra Puzzles. The base models for all models tested come from the Llama 3.2 family of models[2], all experiments were conducted with the 1B and 3B parameter models.

Due to computational constraints, all fine-tuning within this work used LoRA adapters (Hu et al., 2021). In future, testing the same research direction with full fine-tuning would be an interesting direction for continued work.

3.1 Baselines

To evaluate the effect of fine-tuning on synthetic search trajectories, we compare against two baselines: base models and models fine-tuned on reasoning puzzles. If the hypothesis is true, we would expect the search fine-tuned models to outperform the base models and approach the performance of the models directly fine-tuned on the puzzles. The base model is a necessary baseline to check that the search fine-tuning is offering improvement over the model's abilities. Synthetic fine-tuning is not expected to match the performance of the puzzle-fine-tuned models. We hope to see improvements the synthetic fine-tuning so that performance falls between the two baselines.

3.2 Proposed Approach

The proposed models are fine-tuned on the synthetic countdown search trajectories, then performance is measured on each of the two puzzles, Sudoku and Zebra puzzles. The trajectories are generated using the method introduced in Gandhi et al. (2024). For each trajectory, the countdown task is introduced in a user message and the full search trajectory and solution are formatted into the assistant's response. Loss is only calculated on the assistant's message, focusing on the model's ability to create the trajectory and reach a solution. Further details on how each instance is formatted is included in Appendix A.

For each base model, we test performance after training with two quantities of synthetic data, first with 1,000 samples and then increasing to 10,000 samples. Both of the trained models are evaluated in both zero-shot and few-shot contexts (k=3), all results are available in Section 6.

4 Datasets and Evaluation Metrics

4.1 Datasets

To evaluate the effectiveness of our approach, we tested model performance on two datasets of reasoning puzzles: **Sudoku** and **Zebra**. These datasets were selected to test the model's ability to

[2] https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_2/

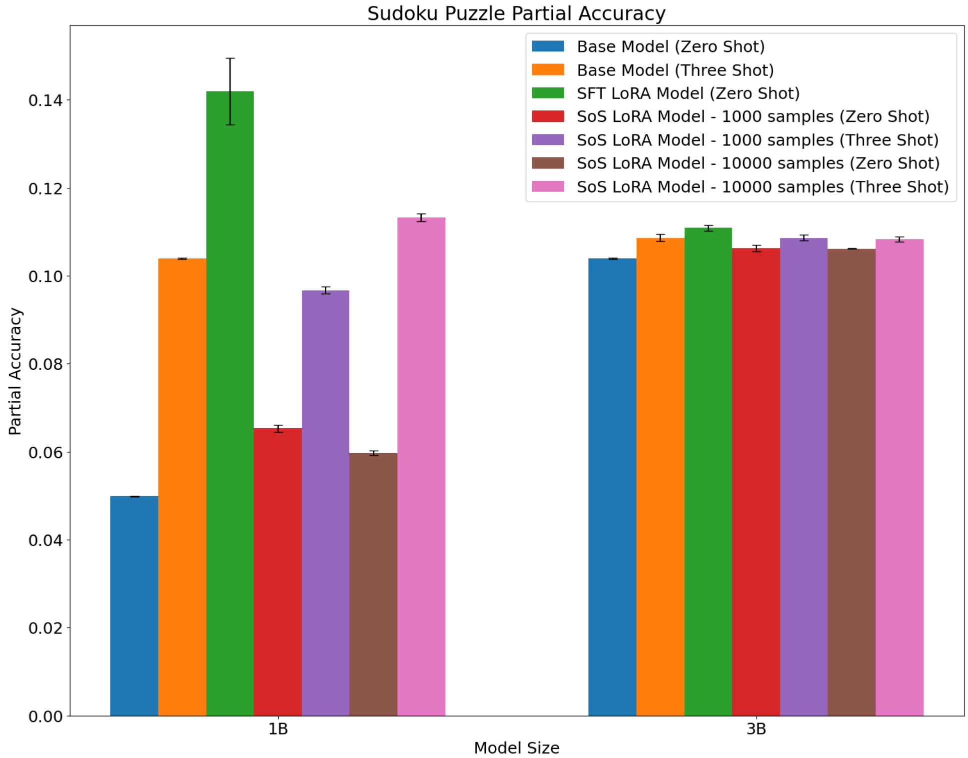

Figure 1: Sudoku Partial Accuracies

perform structured reasoning, arithmetic problem-solving, and logical deduction under varied constraints.

Additionally, we train a model using **Countdown**, a synthetic dataset of search traces. This dataset was used in Gandhi et al. (2024) to improve performance on search tasks.

4.1.1 Sudoku Dataset

The Sudoku dataset focuses on structured logical reasoning within a grid-based numerical framework. Each sample in the dataset contains:

- **Puzzle:** A partially filled 9x9 Sudoku grid, where the task is to complete the grid while ensuring that each row, column, and 3x3 subgrid contains the numbers 1 through 9 without repetition.
- **Clues:** The number of initially provided filled cells, which determines the problem's complexity.
- **Difficulty:** A numeric rating quantifying the puzzle's challenge based on the number of clues and required logical steps for the solution.
- **Solution:** The fully solved Sudoku grid.

The dataset[3] contains 3,000,000 examples, filtered to ensure high-quality reasoning challenges. For training and evaluation, the first 10,000 samples were used, selected for their balanced difficulty levels and diversity of logical patterns.

4.1.2 Zebra Dataset

The Zebra dataset, also known as the "Zebra Puzzle" or "Einstein's Riddle," tests the model's ability to handle deductive reasoning and constraint satisfaction. Each instance presents a set of clues that describe relationships among a collection of entities, such as houses, colors, and owners. The dataset is formatted as:

- **Problem Description:** A set of natural language clues specifying logical constraints (e.g., "The person who owns the zebra lives next to the blue house").
- **Solution:** The complete mapping of entities to attributes that satisfies all constraints.

We used the full 1000 samples available in this online dataset[4].

[3] https://www.kaggle.com/datasets/radcliffe/3-million-sudoku-puzzles-with-ratings/data

[4] https://huggingface.co/datasets/

4.1.3 Countdown Dataset

The Countdown dataset evaluates arithmetic reasoning and problem-solving under constraints. Each sample is a numerical reasoning problem inspired by the classic Countdown game show, where participants manipulate a set of integers using basic arithmetic operations to achieve a target number. Each instance includes:

- **Numbers:** A set of initial integers available for manipulation.
- **Target:** The number to be achieved through a sequence of operations.
- **Solution:** The optimal sequence of arithmetic operations leading to the target.
- **Search Path:** The trajectory explored during solution generation, including intermediate operations and states.
- **Rating:** A score quantifying the quality of the solution path, derived from a heuristic measure of optimality.

Using code[5] from Gandhi et al. (2024), we generated a dataset of 500,000 samples. This dataset was then filtered to retain only trajectories with a rating above 0.995. From this curated subset, we trained two models, the first using 1,000 samples and the second using 10,000 samples.

4.2 Dataset Verbalization

As this work uses instruction-tuned base models, we formatted the datasets into an assistant-user dialog format for supervised fine-tuning. In each case an instance from the dataset would be transformed into the user describing the puzzle and the assistant replying with the solution. Further details on how each dataset was verbalized can be found in Appendix A.

4.3 Evaluation Metrics

To assess the performance of our methods, we utilized two evaluation metrics: **Partial Accuracy** and **Strict Accuracy**. These metrics were applied to the Sudoku and Zebra datasets, enabling comprehensive evaluation of the model's reasoning and problem-solving capabilities across varying levels of granularity.

4.3.1 Partial Accuracy

Partial Accuracy measures the proportion of the solution that is correct, even if the overall solution is incomplete or partially incorrect. This metric is valuable for difficult tasks where models rarely fully solve the puzzle as it provides signal on incremental improvements. For each dataset:

- **Sudoku:** Partial Accuracy evaluates the proportion of correctly filled cells in the 9x9 grid, reflecting the model's understanding of the underlying logical constraints.
- **Zebra:** Partial Accuracy measures the percentage of correctly assigned attributes (e.g., house colors, occupants) relative to the total number of attributes to be assigned.

4.3.2 Strict Accuracy

Strict Accuracy evaluates the correctness of the solution in its entirety. For a sample to achieve strict accuracy, every component of the solution must be correct without any errors or omissions. This metric is critical for assessing the model's ability to provide fully valid and coherent outputs. For each dataset:

- **Sudoku:** Strict Accuracy requires the entire 9x9 grid to be correctly completed, with all rows, columns, and subgrids satisfying the Sudoku constraints.
- **Zebra:** Strict Accuracy demands that the complete mapping of entities to attributes satisfies all given logical constraints.

4.3.3 Rationale for Metric Selection

These metrics were chosen to capture both incremental and holistic reasoning capabilities:

- **Partial Accuracy:** Highlights the model's progress toward solving a problem, even in cases where it fails to achieve the final correct solution.
- **Strict Accuracy:** Ensures that the model is rigorously evaluated on its ability to generate fully correct and valid outputs.

5 Experimental Details

Base model weights were downloaded from Huggingface and the LoRA adapters were implemented

---

allenai/ZebraLogicBench

[5] https://github.com/kanishkg/stream-of-search

with Huggingface's Transformers library[6]. The LoRA adapters targeted modules in the self attention mechanisms of the model. All LoRA adapters were trained with the same configuration: with a rank of 16, an α of 32, a dropout rate of 0.05, and no bias.

Adapters were trained with Supervised Fine-tuning, with loss calculated only on the model's completions. We used the `SFTTrainer` from Huggingface's TRL library[7] which uses the AdamW optimizer with an initial learning rate of 2e-5, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e - 8$.

To improve result reliability, all experiments were run three times. All runs used a random 80-20 train test split so in each experiment run the adapters were trained and tested on different subsets of the datasets.

6 Results and Discussion

6.1 Results

Our experiments evaluated the impact of fine-tuning language models on programmatically generated search traces from the Countdown task, focusing on their ability to generalize to two reasoning tasks, Sudoku and Zebra puzzles. The evaluation metrics used were Partial Accuracy and Strict Accuracy, as described in Section 4 providing complementary insights into intermediate reasoning steps and fully correct solutions, respectively.

Figures 1, 2, and 3 illustrate the performance across different model configurations and training datasets. Appendix B contains full results in a table for more detailed analysis.

6.1.1 Sudoku Partial Accuracy

As seen in Figure 1, none of the models tested models performed well on the Sudoku puzzles. For the 3B models, Partial Accuracies were at or below 11% which is the score you would expect if guessing digits randomly. Surprisingly, the 1B parameter fine-tuned model achieved the highest partial accuracy at around 14%.

6.1.2 Sudoku Strict Accuracy

Strict Accuracy for all configurations in the Sudoku task was consistently zero. This result shows that this puzzle was too difficult for the models tested, even after fine-tuning.

[6] https://huggingface.co/docs/transformers/en/index

[7] https://huggingface.co/docs/trl/en/index

6.1.3 Zebra Puzzle Partial Accuracy

Figure 2 shows that the Zebra fine-tuned model achieved the highest Partial Accuracy, as expected. We find that the base model and SoS-trained models achieve similar performance in most of the trials. We do observe that for the 3B parameter models, the SoS-trained models perform better than the base model in zero shot testing.

6.1.4 Zebra Puzzle Strict Accuracy

We see that the base and SoS fine-tuned models struggled to achieve non-zero Strict Accuracy in the Zebra task, except in a few cases with the 3B model in three-shot settings. The zebra fine-tuned model (SFT LoRA) achieved the best Strict Accuracy scores, indicating its superior ability to generate fully correct solutions. Overall Figure 3 shows that all of the models struggled to produce error-free solutions for Zebra puzzles.

6.2 Discussion

The results highlight critical insights into the strengths and limitations of fine-tuning approaches.

6.2.1 Base Fine-Tuned Model Superiority

Unsurprisingly, across both tasks and metrics, the base fine-tuned model (SFT LoRA) consistently outperformed the SoS fine-tuned models and base models. This finding confirms that directly fine-tuning on task data is very effective and highlights the challenges of eliciting transfer from synthetic data.

6.2.2 Limited Transferability of SoS Fine-Tuning

Fine-tuning on SoS data provided some improvement in Zebra puzzles, particularly in a zero-shot context. These results indicate that the search sub-skill fine-tuned via SoS data did not generalize well to the broader reasoning demands of Zebra and Sudoku tasks.

6.2.3 Impact of Dataset Size, Model Scale and Few-Shot Context

Increasing the number of SoS samples from 1,000 to 10,000 resulted in incremental gains in Partial Accuracy for Zebra puzzles. Few-shot contexts (k=3) further amplified performance improvements for all configurations. Furthermore, larger models (3B) were observed to consistently outperform their smaller counterparts for all configurations.

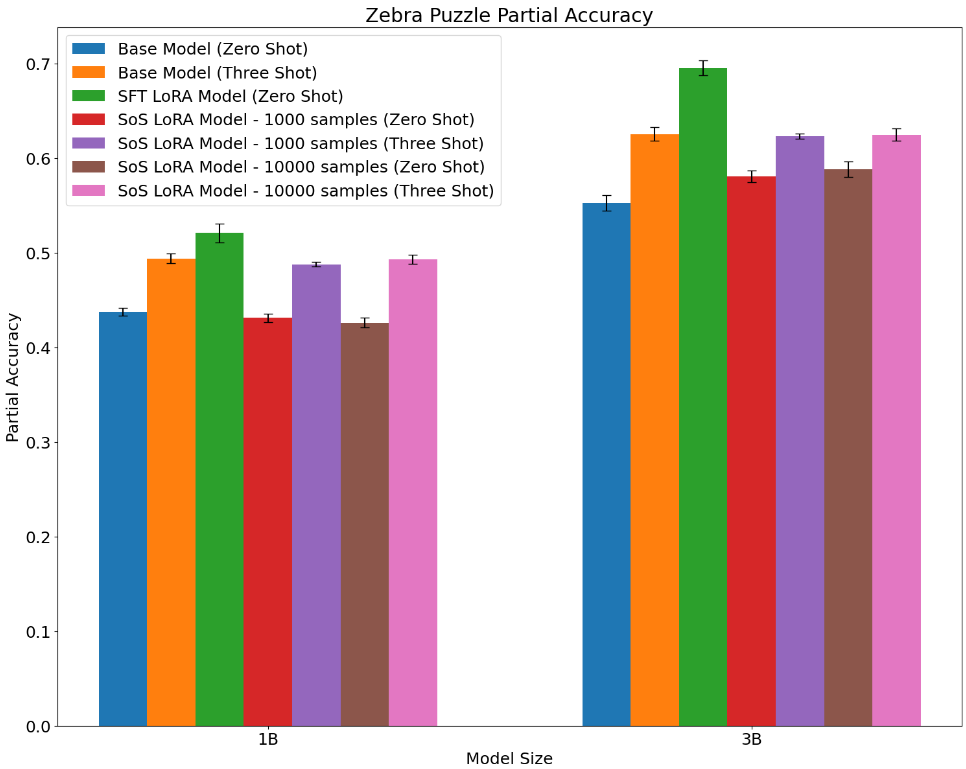

Figure 2: Zebra Partial Accuracies

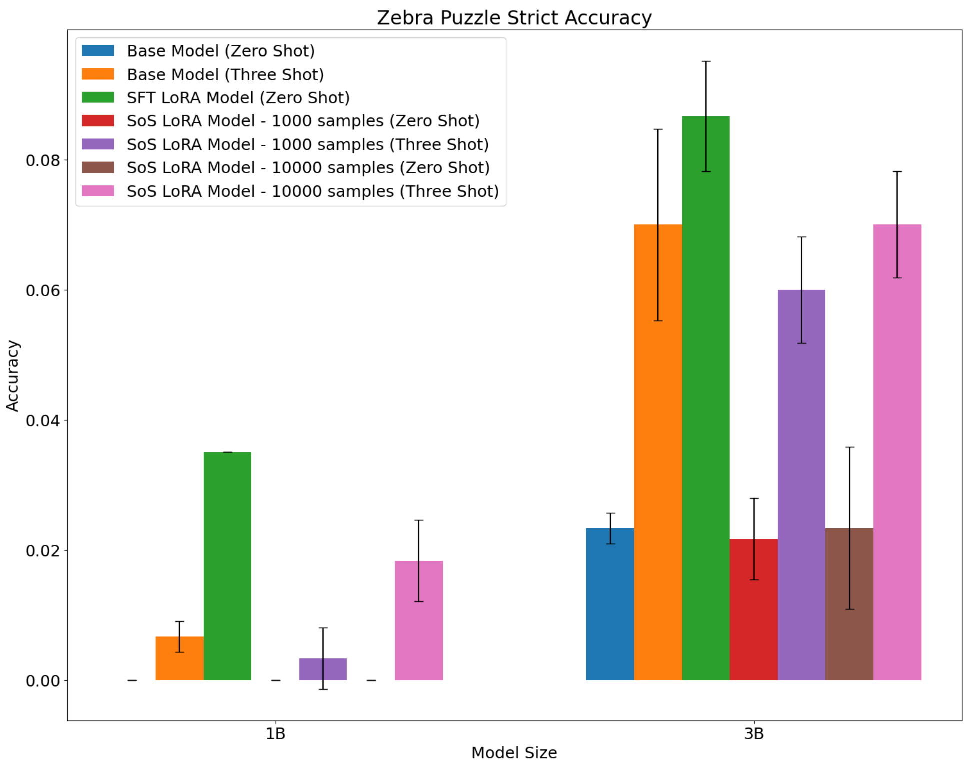

Figure 3: Zebra Strict Accuracies

6.2.4 Task-Specific Challenges

The stark contrast between Sudoku and Zebra results highlights task-specific demands. Zebra puzzles, which rely on natural language reasoning and constraint satisfaction, align more closely with pre-training corpora and thus benefit more from fine-tuning. Poor performance across all models shows that Sudoku was too difficult of a downstream task potentially due to the reliance on structured numerical logic.

7 Conclusion

This work explored the effectiveness of fine-tuning large language models on programmatically generated synthetic data, specifically search trajectories, to enhance reasoning sub-skills and evaluate their impact on downstream tasks such as Sudoku and Zebra puzzles. Our findings demonstrate that while synthetic data fine-tuning provides incremental improvements in Partial Accuracy, especially for Zebra puzzles, its ability to transfer to high-level reasoning tasks is limited. Models fine-tuned directly on task-specific datasets consistently outperformed search-fine-tuned models across all metrics and tasks, emphasizing the difficulty in synthetic data approaches. Sudoku was revelaed to be too difficult of a task for the models tested as evidenced by Strict Accuracies of 0%. Larger models (3B parameters) and few-shot contexts improved performance, but the observed gains remained task-specific; Zebra puzzles benefited more possibly due to their alignment with natural language reasoning, whereas Sudoku posed a greater challenge. These results highlight the potential of synthetic data for training foundational reasoning sub-skills but suggest that combining it with targeted task-specific fine-tuning may be beneficial in the future.

7.1 Future Work

In the future, work can be done to explore hybrid approaches, expanded reasoning sub-skills, and scaling model size to enhance generalization and improve reasoning performance across diverse tasks. Furthermore, it would be interesting to repeat a similar set of experiments with more reasoning sub-skills such as symbolic reasoning or arithmetic. Expanding the number of sub-skills would allow for analysis of how different sub-skills transfer to different downstream reasoning tasks. Additionally, we would like to increase the size of the base models tested since it is possible that the 1B and 3B parameter models used in this work don't have the capacity to learn generalizable sub-skills but that ability may emerge at larger scales.

8 Contributions

Cormac developed the training and evaluation scripts and documented the repository for release. Abhijeet handled the synthetic data generation and built all of the data loaders. Writing the report was a joint effort.

Acknowledgments

Thank you to Xiaoyin Chen for the initial research direction, continued mentorship throughout the project, and providing access to compute resources. Thank to Arkil Patel for valuable feedback on our project proposal. Thank you to Prof. Siva Reddy for an excellent LLMs course and being available as a resource throughout the term.

References

Hritik Bansal, Arian Hosseini, Rishabh Agarwal, Vinh Q. Tran, and Mehran Kazemi. 2024. Smaller, weaker, yet better: Training llm reasoners via compute-optimal sampling.

Xiang Deng, Yu Su, Alyssa Lees, You Wu, Cong Yu, and Huan Sun. 2021. ReasonBERT: Pre-trained to reason with distant supervision. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6112–6127, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. 2024. Stream of search (sos): Learning to search in language.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-star: Training verifiers for self-taught reasoners.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685.

Fangkai Jiao, Yangyang Guo, Xuemeng Song, and Liqiang Nie. 2022. Merit: Meta-path guided contrastive learning for logical reasoning.

Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul Mcvay, Michael Rabbat, and Yuandong Tian. 2024. Beyond a*: Better planning with transformers via search dynamics bootstrapping.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's Verify Step by Step. ArXiv:2305.20050.

Xinyu Pi, Qian Liu, Bei Chen, Morteza Ziyadi, Zeqi Lin, Qiang Fu, Yan Gao, Jian-Guang Lou, and Weizhu Chen. 2022. Reasoning like program executors.

Kulin Shah, Nishanth Dikkala, Xin Wang, and Rina Panigrahy. 2024. Causal language modeling can elicit search and reasoning capabilities on logic puzzles.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. 2024. Beyond human data: Scaling self-training for problem-solving with language models.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024. Chain of thoughtlessness? an analysis of cot in planning.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large language models still can't plan (a benchmark for LLMs on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. ArXiv:2203.11171.

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.

Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2022. Linkbert: Pretraining language models with document links.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. 2024. Physics of language models: Part 2.2, how to learn from mistakes on grade-school math problems.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1.

A Dataset Verbalization Format

A.1 Sudoku

Sudoku Dataset Verbalized Input

[User]: Given the Sudoku puzzle 1..5.37..6.3..8.9......98...1.......8761..........6...... .....7.8.9.76.47...6.312, which has 27 clues and a difficulty rating of 2.2. Please solve for the final arrangement.

Sudoku Dataset Verbalized Output

[Assistant]: 198543726643278591527619843 914735268876192435235486179 462351987381927654759864312

A.2 Zebra

Zebra Dataset Verbalized Input

[User]: Given there are 2 houses, numbered 1 to 2 from left to right. Each house is occupied by a different person. Each house has a unique attribute for each of the following characteristics:
- Each person has a unique name: Arnold, Eric
- People own unique car models: ford f150, tesla model 3
- The people keep unique animals: cat, horse

Clues:
1. Eric is directly left of the person who owns a Tesla Model 3.
2. The person who keeps horses is in the first house.

Please solve for the final arrangement.

Zebra Dataset Verbalized Output

[Assistant]: The solution is as follows:
In house 1, name is Eric, car model is ford f150, and animal is horse.
In house 2, name is Arnold, car model is tesla model 3, and animal is cat.

A.3 Countdown

Countdown Dataset Verbalized Input

[User]: You are tasked to solve arithmetic reasoning problems. Given a set of numbers and a target, describe the steps in the path to reach the target using those numbers. Using arithmetic operations such as addition (+), subtraction (-), multiplication (*) and division (/), use the initial set of numbers to gather new numbers that eventually reach the target in the end. The target is: 48 The available numbers are: [12, 96, 68, 32] Describe how to reach the target using the given numbers.

Countdown Dataset Verbalized Output

[Assistant]: The search path used for this problem was: Current State: 48:[12, 96, 68, 32], Operations: [] Exploring Operation: 96-68=28, Resulting Numbers: [12, 32, 28] Generated Node #0,0: 48:[12, 32, 28] Operation: 96-68=28 Moving to Node #0,0 Current State: 48:[12, 32, 28], Operations: ['96-68=28'] Exploring Operation: 32-28=4, Resulting Numbers: [12, 4] Generated Node #0,0,0: 48:[12, 4] Operation: 32-28=4 Moving to Node #0,0,0 Current State: 48:[12, 4], Operations: ['96-68=28', '32-28=4'] Exploring Operation: 12+4=16, Resulting Numbers: [16] 16,48 unequal: No Solution Moving to Node #0,0,0 Current State: 48:[12, 4], Operations: ['96-68=28', '32-28=4'] Exploring Operation: 12*4=48, Resulting Numbers: [48] 48,48 equal: Goal Reached
The optimal path was: Current State: 48:[12, 96, 68, 32], Operations: [] Exploring Operation: 96-12=84, Resulting Numbers: [68, 32, 84] Generated Node #2: [68, 32, 84] from Operation: 96-12=84 Current State: 48:[68, 32, 84], Operations: ['96-12=84'] Exploring Operation: 68-32=36, Resulting Numbers: [84, 36] Generated Node #3: [84, 36] from Operation: 68-32=36 Current State: 48:[84, 36], Operations: ['96-12=84', '68-32=36'] Exploring Operation: 84-36=48, Resulting Numbers: [48] 48,48 equal: Goal Reached
The final solution was: 96-12=84, 68-32=36, 84-36=48

B Results Tables

B.1 Full Sudoku Results

Run Configuration	**Partial Accuracy**	**Strict Accuracy**
Base Evaluation Runs		
Zero-Shot, 1B Model	0.0499 (0.00004)	0.0 (0.0)
Three-Shot, 1B Model	0.104 (0.0001)	0.0 (0.0)
Zero-Shot, 3B Model	0.104 (0.0001)	0.0 (0.0)
Three-Shot, 3B Model	0.109 (0.0008)	0.0 (0.0)
Fine-Tuned Training Runs		
Zero-Shot, 1B Model	0.111 (0.008)	0.0 (0.0)
Zero-Shot, 3B Model	0.142 (0.0006)	0.0 (0.0)
SOS Evaluation - 1000 Samples		
Zero-Shot, 1B Model	0.065 (0.0008)	0.0 (0.0)
Three-Shot, 1B Model	0.097 (0.0008)	0.0 (0.0)
Zero-Shot, 3B Model	0.106 (0.0007)	0.0 (0.0)
Three-Shot, 3B Model	0.109 (0.0007)	0.0 (0.0)
SOS Evaluation - 10000 Samples		
Zero-Shot, 1B Model	0.059 (0.0005)	0.0 (0.0)
Three-Shot, 1B Model	0.113 (0.0008)	0.0 (0.0)
Zero-Shot, 3B Model	0.106 (0.00008)	0.0 (0.0)
Three-Shot, 3B Model	0.108 (0.0006)	0.0 (0.0)

B.2 Full Zebra Puzzle Results

Run Configuration	Partial Accuracy	Strict Accuracy
Base Evaluation Runs		
Zero-Shot, 1B Model	0.438 (0.004)	0.0 (0.0)
Three-Shot, 1B Model	0.4947 (0.005)	0.007 (0.002)
Zero-Shot, 3B Model	0.553 (0.008)	0.023 (0.002)
Three-Shot, 3B Model	0.625 (0.007)	0.07 (0.014)
Fine-Tuned Training Runs		
Zero-Shot, 1B Model	0.521 (0.010)	0.035 (0.0)
Zero-Shot, 3B Model	0.695 (0.008)	0.087 (0.008)
SOS Evaluation - 1000 Samples		
Zero-Shot, 1B Model	0.431 (0.005)	0.0 (0.0)
Three-Shot, 1B Model	0.488 (0.002)	0.003 (0.005)
Zero-Shot, 3B Model	0.581 (0.006)	0.022 (0.006)
Three-Shot, 3B Model	0.623 (0.003)	0.06 (0.008)
SOS Evaluation - 10000 Samples		
Zero-Shot, 1B Model	0.426 (0.005)	0.0 (0.0)
Three-Shot, 1B Model	0.493 (0.005)	0.018 (0.006)
Zero-Shot, 3B Model	0.588 (0.008)	0.023 (0.012)
Three-Shot, 3B Model	0.625 (0.007)	0.07 (0.008)