Improving Mathematical Reasoning in LLMs

DA322 Course Project

Himanshu Singhal
220150004
3rd Year BTech, DSAI
IIT Guwahati

Rishita Agarwal
220150016
3rd Year BTech DSAI
IIT Guwahati

Arushi Kumar
220150032
3rd Year Btech DSAI
IIT Guwahati

Raunit Patel
220150015
3rd Year Btech DSAI
IIT Guwahati

***Abstract*—Mathematical reasoning encompasses a broad spectrum of abilities, ranging from the execution of simple arithmetic operations to solving complex problems requiring deep logical inference and symbolic manipulation. Recent advancements in artificial intelligence have aimed to automate and enhance the capability of machines to understand and solve such mathematical tasks, which are pivotal in fields ranging from theoretical computer science to practical engineering applications. This paper presents a comparative study of four distinct computational approaches that address the challenge of mathematical reasoning in automated systems: Prompting, EntropiX, Monte Carlo Tree Search (MCTS), and Neurosymbolic computing.**

We measure performance based on accuracy, efficiency (time to solution), and scalability. Our findings suggest that while no single approach outperforms the others across all metrics, the choice of method can be optimized based on specific problem characteristics and requirements. This study not only advances our understanding of AI-driven mathematical problem solving but also highlights significant directions for future research in combining these methodologies to leverage their respective advantages.

I. Literature Review

A. Introduction to Mathematical Reasoning in AI

Mathematical reasoning involves the capacity to understand, conceptualize, and solve problems using mathematical concepts. In artificial intelligence, the challenge has been to develop systems that can emulate or even surpass human capabilities in this domain. As observed by Lample and Charton (2020), the journey towards achieving proficiency in mathematical reasoning has gone through various phases of AI development, beginning with rule-based systems and extending into the realm of advanced machine learning techniques.

LLM struggles with mathematical reasoning in following ways:

- LLM might provide different answers upon repeated queries or offer solutions that are not consistently correct. This inconsistency can be particularly problematic in educational or professional settings where accuracy is critical.
- They are better at mimicking the form of mathematical reasoning seen in their training data rather than genuinely understanding underlying mathematical principles. This limitation becomes apparent in more complex or abstract mathematical domains, such as higher mathematics involving advanced calculus, proofs, or complex algebraic structures.
- While LLMs can generate solutions that appear correct, they often cannot provide a step-by-step explanation or justification in a way that aligns with mathematical rigor. This lack of explainability is a significant drawback in educational contexts where understanding the process is as important as the answer.

B. Large Language Models and Prompting Techniques

The emergence of large language models (LLMs) like GPT-3 (Brown et al., 2020) has redefined the boundaries of what machines can achieve in understanding and generating human-like text, including solving mathematical problems. The use of prompting techniques to guide these models in producing structured mathematical proofs or solutions has been documented extensively (Chen et al., 2021). These studies demonstrate that when properly primed, LLMs can generate coherent and logically sound mathematical arguments. However, issues related to consistency and reliability remain, as these models often generate correct and incorrect solutions with similar confidence.

C. Entropy Minimization in Mathematical

Optimization Entropy minimization approaches, as described by Singh and Gupta (2019), leverage the concept of reducing disorder within a system to improve decision-making accuracy. EntropiX, an adaptation of this principle, focuses on refining algorithms' ability to navigate through complex problem-solving processes by iteratively reducing uncertainty. This technique has been particularly useful in structured environments where prior knowledge about the problem space can significantly enhance performance.

D. Monte Carlo Tree Search (MCTS)

The adaptation of Monte Carlo Tree Search (MCTS) for mathematical problem-solving was explored in-depth by

Browne et al. (2012). MCTS has been effective in complex decision-making environments, such as games, where a balance between exploration of new paths and exploitation of known paths is crucial. The application of MCTS in mathematical reasoning, as detailed by Silver et al. (2017) in the development of AlphaGo, has shown that this approach can effectively handle the intricacies of strategic problem-solving in mathematics.

MCTS step by step-

- **Selection**: Starting at the root node, the tree policy navigates through the tree to a leaf node using a specific strategy that balances exploration of unvisited nodes and exploitation of nodes known to yield good outcomes.
- **Expansion**: Unless the leaf node ends the game or decision process, it adds one or more child nodes to the tree, expanding the search space.
- **Simulation**: From the new nodes, it simulates random outcomes of the decision process (often called the rollout phase).
- **Backpropagation**: It then updates the nodes with new statistical information about the potential of that node leading to a win or desirable outcome.

E. Neurosymbolic Computing

Neurosymbolic computing seeks to bridge the gap between neural network-based learning and symbolic reasoning. Garcez et al. (2020) provide a comprehensive overview of how integrating these two paradigms can offer robust solutions to problems that require both pattern recognition and logical deduction. This dual approach facilitates not only the processing of numerical data but also the understanding and manipulation of symbolic mathematical expressions, thereby enhancing the interpretability and scalability of solutions.

II. Practical Applications

A. Neurosymbolic AI for Visual Question Answering

The "Neuro-Symbolic AI for Visual Question Answering" provides an implementation of a **Neuro-Symbolic AI (NSAI) model** for Visual Question Answering (VQA) tasks using the **Sort-of-CLEVR dataset**.

The core idea of the project is to combine Neural Networks' pattern recognition capabilities with symbolic reasoning. This hybrid approach leverages the strengths of both deep learning (for visual recognition) and symbolic methods (for logical reasoning and program execution).

1) ***Key components****:*

- **Neuro-Symbolic AI** integrates the power of deep learning with symbolic systems, enabling better reasoning over visual data.
- The **Sort-of-CLEVR** dataset is used for the **VQA task**, which involves answering questions about images in a synthetic environment.
- The visual input (image) is parsed into a **symbolic** representation. This symbolic representation is then processed to interpret the visual elements in a way that a symbolic reasoner can use.
- The query (usually a **natural language question**) is parsed into an executable program. This program will be executed on the symbolic representation of the image to derive the answer.
- The **program executor** takes the parsed query and executes it on the symbolic scene representation to determine the answer. The architecture is built around program synthesis, where a **symbolic** program is dynamically generated from the input query.

2) ***High-Level Architecture****:* The implementation follows the workflow:

- **Perception Module**: Detects and interprets the objects and attributes from the image.
- **Semantic Parser**: Converts the natural language query into a logical program.
- **Program Executor**: Executes the program on the symbolic representation of the scene to answer the question.

B. Chess with Monte Carlo Tree Search & CNN

This application focuses on developing an AI that plays chess by leveraging the Monte Carlo Tree Search (MCTS) algorithm, enhanced by Convolutional Neural Networks (CNN). MCTS is used for decision-making and move exploration, while CNNs evaluate board positions. The AI employs a self-play method to train and improve through experience, optimizing gameplay strategies without the need for a preprogrammed strategy.

1) ***Key Components of the Project****:*

- **Monte Carlo Tree Search (MCTS**:
  - MCTS is a search algorithm used to make decisions by simulating multiple possible future moves in the game.
  - The search is structured as a tree where each node represents a game state, and edges represent possible moves.
  - The algorithm evaluates possible future game states by running simulations and selecting moves based on statistical outcomes (win/loss/draw).
  - MCTS explores the game tree by expanding nodes that are statistically most likely to lead to a winning position, balancing exploration and exploitation using the UCT (Upper Confidence Bound applied to Trees) formula.
  - This project specifically customizes the MCTS to optimize its performance in chess by adjusting parameters like the exploration constant.
- **Convolutional Neural Networks (CNN)**
  - CNNs are employed to evaluate board positions and predict the best possible moves.
  - The CNN model is designed to take the current board state as input and output predictions for the best move.
  - CNN layers are structured to understand spatial dependencies (e.g., piece locations, their interactions)

that are critical for evaluating a chessboard effectively.
  - The network is trained on data generated from games played between the AI itself, reinforcing successful strategies and moves.
- **Convolutional Neural Networks (CNN)**
  - The model has 2 branches, one for the board and one for the turn.
    - The board branch is a CNN that takes in the board as a tensor and outputs a tensor.
    - The turn branch is a fully connected neural network that takes in the turn as a tensor and outputs a tensor.
  - The two branches are then merged and passed through a series of fully connected layers. The output layer is a fully connected ndoes, one for each possible result. The model is compiled with the Adam optimizer, categorical crossentropy loss etc.
  - The model is trained for **128 epochs** with a **batch size of 64** and a validation split of **0.3**.
- **Self-Play and Reinforcement Learning**
  - The system improves itself through self-play, where the AI plays against itself to accumulate training data.
  - The training uses Reinforcement Learning (RL), where the AI adjusts its strategy based on the results of games (win/loss).
  - During self-play, the AI alternates between exploring new, untried moves and exploiting known successful moves. This exploration-exploitation balance is crucial for preventing the AI from settling into suboptimal strategies.
  - Data generation and model training are continuous, with each game adding to the dataset, enhancing the network's ability to predict optimal moves.
- **Game State Representation**
  - Chessboard positions are encoded into a format suitable for input into the CNN. Each square of the chessboard is represented as a 17-dimensional vector that encodes information about the type of piece, castling rights, and whose turn it is. This 17-dimensional representation is crucial for the network's understanding of the game state.
  - This data representation allows the CNN to better process the game's spatial and temporal dependencies, which are necessary for strategic decision-making.

III. Results and Experimental examples

A. Neuro Symbolic

1) Neuro Symbolic algorithm experimentation: We are given the following information:

- Janet's ducks lay a total of 16 eggs per day, i.e., $D = 16$.
- Janet eats 3 eggs for breakfast every morning, i.e., $B = 3$.

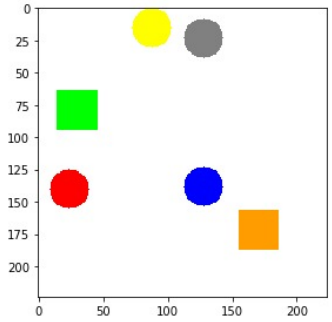

Fig. 1. Input Image

- Janet uses 4 eggs to bake muffins for her friends every day, i.e., $M = 4$.
- Janet sells the remaining eggs at the farmers' market for \$2 per egg, i.e., $P = 2$.

We are asked to calculate how much Janet makes every day at the farmers' market.

Step 1: Calculate the total number of eggs consumed and used

Janet consumes and uses a total of:

$$B + M = 3 + 4 = 7$$

Step 2: Calculate the number of eggs remaining to be sold

The remainder of the eggs, which are sold at the farmers' market, is:

$$R = D - (B + M) = 16 - 7 = 9$$

Step 3: Calculate the total earnings from selling the remaining eggs

The total earnings from selling the remainder of the eggs can be calculated by multiplying the number of eggs sold (R) by the price per egg (P):

$$E = R \times P = 9 \times 2 = 18$$

Conclusion: Janet makes a total of \$18 every day at the farmers' market.

2) Neuro Symbolic AI for VQA: The fig. 1 is input image on which the following questions are asked.

Queries and Answers

The following is a subset of **training data** -

1) **Question:** What is the closest shape to the orange object?
   Answer: Circle
   Reasoning: filter orange ⟨ nxt ⟩ relate closest ⟨ nxt ⟩ query shape
2) **Question:** What is the furthest shape from the yellow object?
   Answer: Rectangle

Reasoning: filter yellow ⟨ nxt ⟩ relate furthest ⟨ nxt ⟩ query shape

3) **Question:** What is the furthest shape from the yellow object?
Answer: Rectangle
Reasoning: filter yellow ⟨ nxt ⟩ relate furthest ⟨ nxt ⟩ query shape
4) **Question:** What is the furthest shape from the green object?
Answer: Rectangle
Reasoning: filter green ⟨ nxt ⟩ relate furthest ⟨ nxt ⟩ query shape
5) **Question:** What is the furthest shape from the green object?
Answer: Rectangle
Reasoning: filter green ⟨ nxt ⟩ relate furthest ⟨ nxt ⟩ query shape

The **testing data** looks like this -

1) **Question:** What is the closest shape to the orange object?
Answer: Circle
2) **Question:** What is the furthest shape from the yellow object?
Answer: Rectangle
3) **Question:** What is the furthest shape from the yellow object?
Answer: Rectangle
4) **Question:** What is the furthest shape from the green object?
Answer: Rectangle
5) **Question:** What is the furthest shape from the green object?
Answer: Rectangle
6) **Question:** What is the furthest shape from the gray object?
Answer: Circle
7) **Question:** How many objects of the same shape as the red object are there?
Answer: 4
8) **Question:** What is the closest shape to the red object?
Answer: Rectangle
9) **Question:** What is the closest shape to the green object?
Answer: Circle
10) **Question:** How many objects of the same shape as the blue object are there?
Answer: 4

CONCLUSION

The model in **Neuro Symbolic AI for VQA** achieves over **99 percent** accuracy on the **Sort-of-CLEVR dataset**, which is impressive given the complexity of combining visual and linguistic inputs.

B. Prompting

Example 1: Question: If you buy 5 apples costing $2 each, how much do you spend in total?
Answer:

- First, determine the cost of one apple, which is $2.
- Since you buy 5 apples, multiply the number of apples by the cost per apple:

$$5 \text{ apples} \times 2 \text{ dollars/apple} = 10 \text{ dollars}.$$

- **Total expenditure:** $10.

Example 2: Question: A rectangle has a length of 10 meters and a width of 5 meters. What is its area?
Answer:

- The area of a rectangle is calculated by multiplying the length by the width.
- The length is 10 meters, and the width is 5 meters.
- Area calculation:

$$10 \text{ meters} \times 5 \text{ meters} = 50 \text{ square meters}.$$

- **Area:** 50 square meters.

Example 3: Question: If you have a $50 bill and you buy a toy for $15, how much change should you receive?
Answer:

- First, determine the cost of the toy, which is $15.
- Subtract the cost of the toy from the amount you have:

$$50 \text{ dollars} - 15 \text{ dollars} = 35 \text{ dollars}.$$

- **Change received:** $35.

C. Entropix Results

Accuracy of approx **26.6 percent** was observed on various queries.

REFERENCES

[1] Brown, T. B., et al. (2020). "Language Models are Few-Shot Learners." OpenAI.
[2] Chen, X., et al. (2021). "Evaluating Large Language Models Trained on Code." arXiv.
[3] Lample, G., Charton, F. (2020). "Deep Learning for Symbolic Mathematics." arXiv.
[4] Garcez, A., et al. (2020). "Neurosymbolic AI: The 3rd Wave." arXiv.
[5] Singh, S., Gupta, A. (2019). "Entropy Minimization in Deep Learning." arXiv.
[6] Browne, C., et al. (2012). "A Survey of Monte Carlo Tree Search Methods." IEEE Transactions on Computational Intelligence and AI in Games.
[7] Silver, D., et al. (2017). "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm." arXiv.
[8] https://arxiv.org/pdf/2405.18357 - Faithful Reasoning Through Symbolic Chain of Thought
[9] https://arxiv.org/pdf/2407.20311 - Physics of LLMs (deep dive if they actually reason)
[10] https://arxiv.org/pdf/2406.07394v2 - Monte Carlos Trees Self Refine
[11] K. Elissa, "Title of paper if known," unpublished.
[12] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.
[13] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[14] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

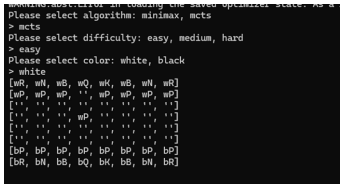

Fig. 2.

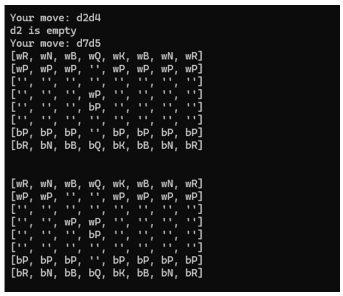

Fig. 3.

APPENDIX

MCTS and chess game play experiment

The following are the steps of the game play in the mcts chess - The figures from 2 to 11 indicate steps.

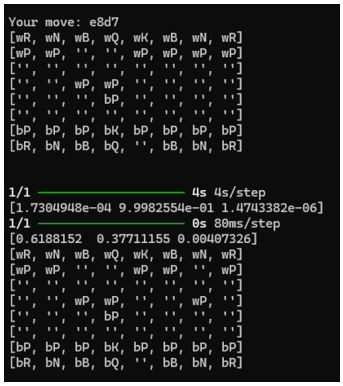

Fig. 4.

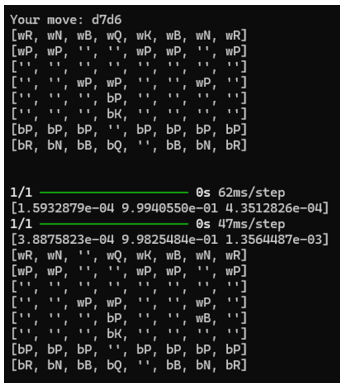

Fig. 5.

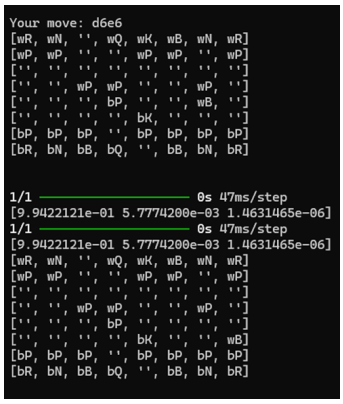

Fig. 6.

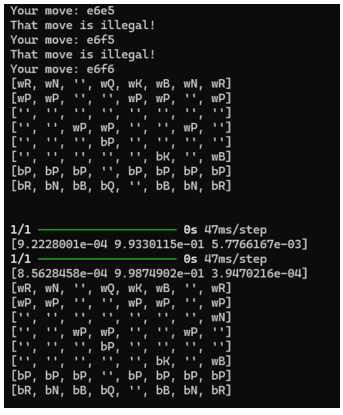

Fig. 7.

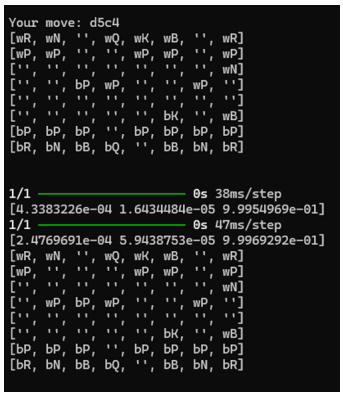

Fig. 8.

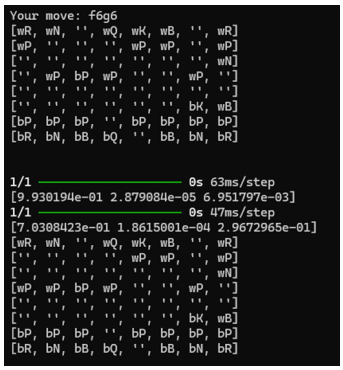

Fig. 9.

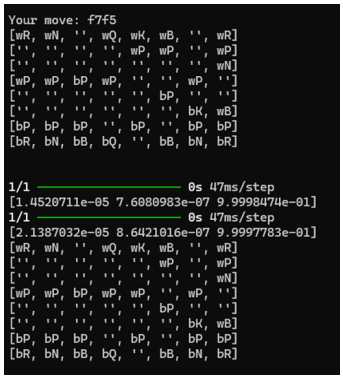

Fig. 10.

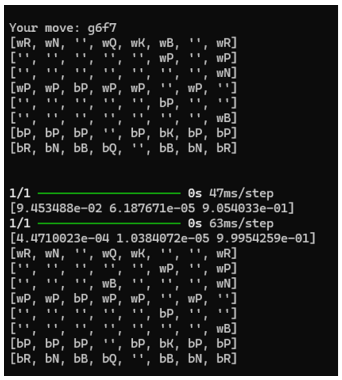

Fig. 11.