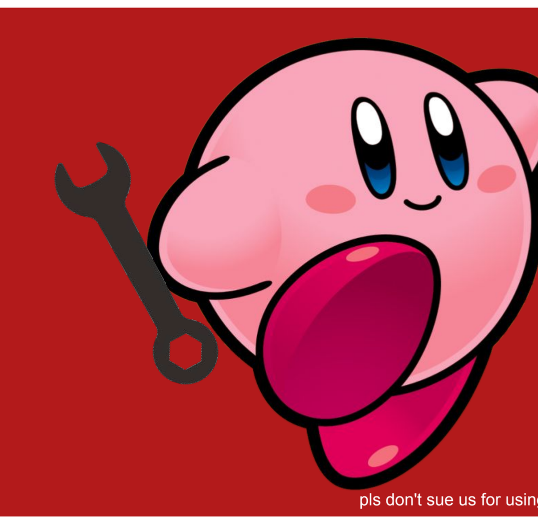
Tree of Thoughts

Sophia Baik, Nicole Ku, William Leung, Ashley Son

Cornell Ann S. Bowers College of Computing and Information Science



pls don't sue us for using copyrighted images

INTRODUCTION

- Traditional LLMs reason linearly and lack structured planning, limiting performance on complex tasks.
- **Tree of Thoughts (ToT)** enables more deliberate problem solving by generating and evaluating intermediate **"thoughts"** and exploring multiple solution paths.
- **Our Goal**: enhance LLM accuracy on arithmetic problems using the Game of 24
- **Game of 24 (G24):** Combine four numbers with basic operations to make 24.
  - [2, 3, 5, 12] -> (12 / (3 - 5/2))

CONCLUSION

- ToT outperforms traditional LLMs on structured reasoning tasks like G24
- **Backtracking** and **systematic exploration** are key to performance gains.
- Future work:
  - Adaptive ToT: decide when to use ToT based on LLM certainty
  - Generalization: apply ToT to other games (Sudoku, word ladders)

REFERENCES

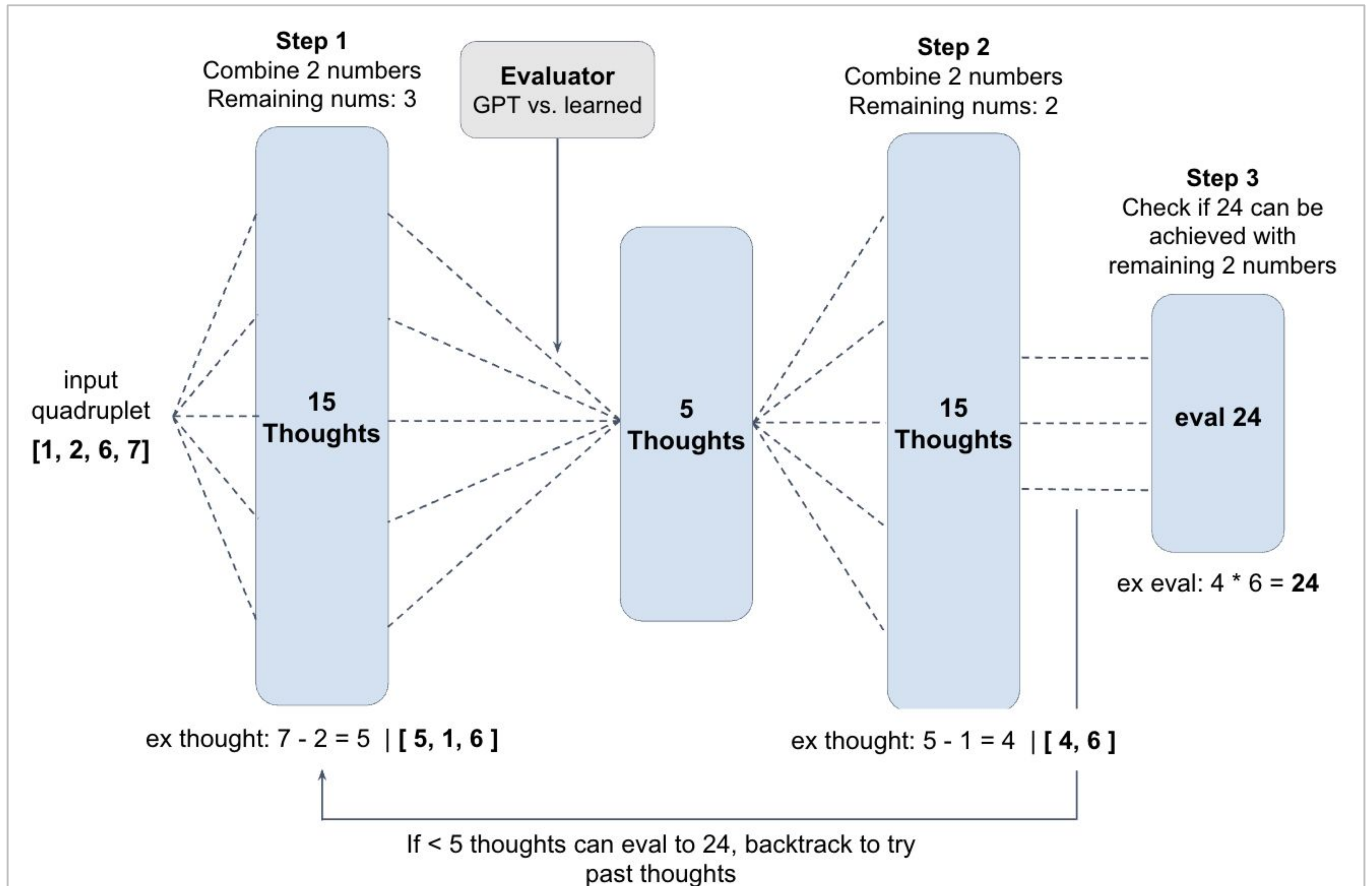
[1] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. 2023.

[2] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems, 32, 8026–8037.

METHODOLOGY

- **Data:** 1,362 solvable 4 number combinations, categorized into 5 difficulty levels, based on the number of possible solutions.
- **Control:** GPT 4o-mini with 5-shot prompting
- **ToT algorithm:**
  1. Use GPT to generate b × 3 ways to combine 2 of the 4 numbers. Use an evaluator to prune to the top b thoughts.
  2. From the 3 remaining numbers, generate b × 3 new combinations of selecting 2 numbers and prune to b
  3. GPT combines the final 2 numbers and checks if the result is 24.
  4. **Backtracking (optional):** If there are no valid final expressions, revisit unused thoughts from Step 1.
- **Evaluators:**
  1. LLM evaluator: GPT rates states as sure/likely/impossible
  2. Learned Evaluator: Neural network trained to estimate value V(state) from features.

Figure 1: ToT architecture



RESULTS

- **Accuracy:** All ToT models outperform 5-shot GPT at all difficulties
  - Backtracking yields accuracy comparable to the original paper
  - LLM evaluator > Learned evaluator > 5-shot
- **Cost:** ToT with LLM evaluator model is **150x** cheaper than the original paper.
- **Datasets:** We had 2 different approaches.
  - Table 1 uses the paper's dataset, which includes a subset of 100 problems of around 2.88 difficulty.
  - Figure 2 uses our dataset. We performed batches of testing on 10 random G24 problems for each difficulty level.

Figure 2: Accuracy by Difficulty Level

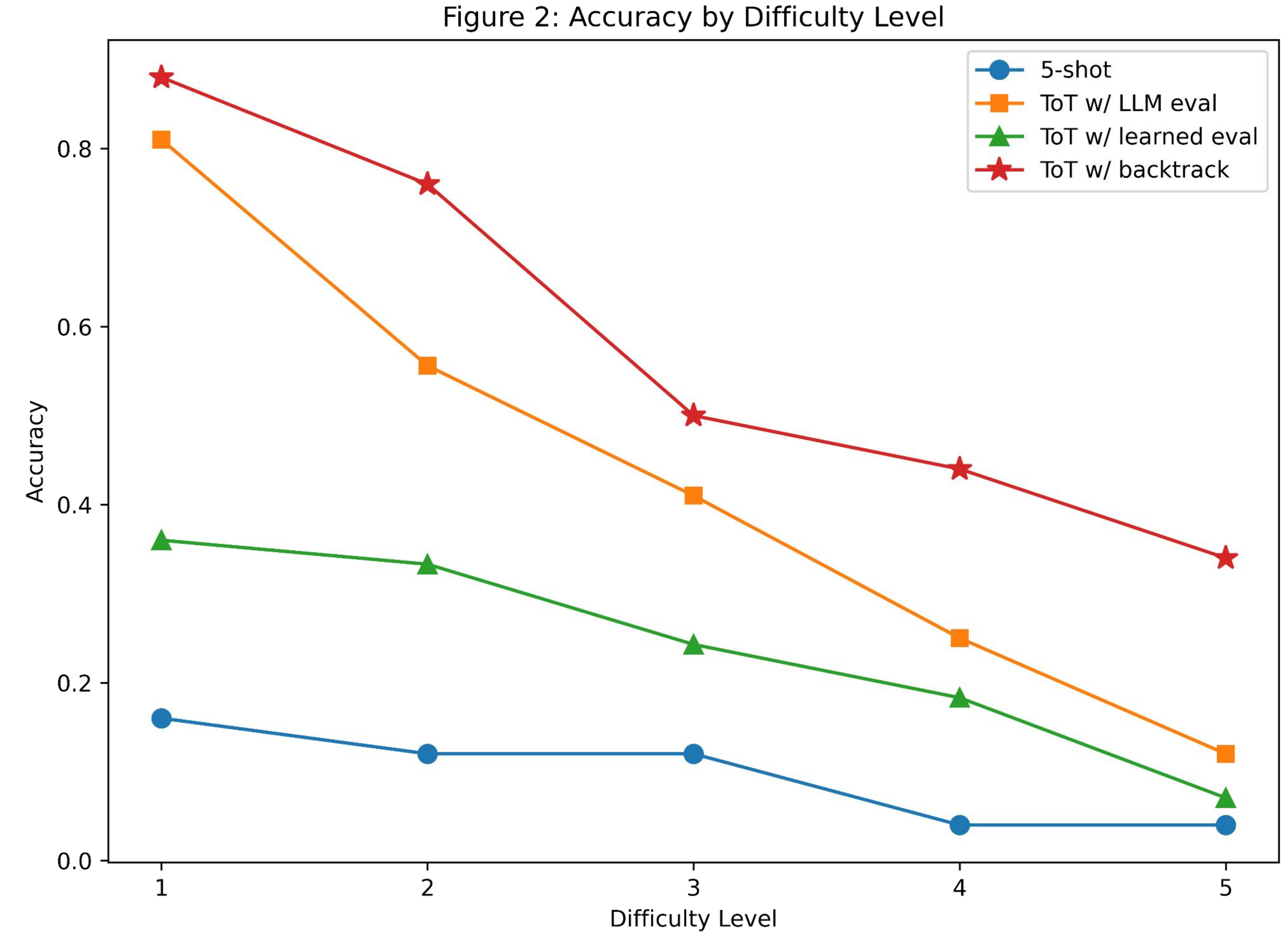


Table 1: Metric Comparisons

Method	Accuracy	Cost per problem	Time per problem
Paper's ToT	**0.74**	$0.74	–
Our ToT w/ Backtracking	0.68	$0.0064	81.69 secs
Our ToT w/ LLM Eval	0.49	$0.0049	60.62 secs
Our ToT w/ Learned Eval	–	**$0.0020**	**23.53 secs**