Tiny Recursive Models vs. Frontier Foundational Models Performance on nanogram Puzzles

Team Members: Keyan Mikaili, Andre Keyser, Samik Bhinge

Roles

- **Keyan:** Model architecture and posttraining
- **Andre:** Data Loading and contextual model application
- **Samik:** Data engineering and pretraining

Problem Description

Nonograms (also known as Picross or Griddlers) are logic puzzles requiring constraint satisfaction to determine which cells in a grid should be filled based on numeric clues. Classical algorithms using line-solving heuristics and backtracking achieve near-perfect accuracy and solve puzzles in seconds. However, recently, Tiny Recursive Models (TRMs) have demonstrated that small neural networks (7M parameters) using recursive reasoning can achieve competitive performance on structured reasoning tasks like ARC-AGI and Sudoku with significantly fewer parameters than large language models.

This project will conduct a systematic comparison between classical constraint satisfaction solvers and neural recursive reasoning on a standardized nonogram benchmark. The primary research question is: To what extent does learned recursive reasoning compare to algorithmic constraint satisfaction on nonogram puzzles, and what are the fundamental tradeoffs between these approaches?

This study will address this question by implementing a TRM architecture following the approach from "Less is More: Recursive Reasoning with Tiny Networks" (Jolicoeur-Martineau, 2025) and comparing its performance, efficiency, and failure modes against established classical solvers. The core challenge is to evaluate if neural approaches can replicate the precision of classical solvers while offering distinct advantages in noisy environments or transfer learning contexts.

Secondary objectives include:

- Analyzing the sample efficiency of TRM vs. classical approaches (how much training data is required for competitive performance).
- Identifying problem characteristics where neural approaches might offer advantages (e.g., noisy inputs, puzzle variants, transfer learning potential).
- Quantifying the computational cost tradeoffs (training time, inference speed, memory requirements).

Preliminary Plan

1. Dataset Preparation & Baseline Implementation

- **Objective:** To establish a robust evaluation framework with classical solver baselines.
- **Tasks:**
    - Download and preprocess 8,000+ puzzles from webpbn.com with varying difficulty levels.
    - Implement classical line-solving algorithm with backtracking as primary baseline.
    - Create standardized train (60%), validation (20%), and test (20%) splits.
    - Establish baseline metrics: solve rate, solve time, and number of search nodes explored.

2. TRM Architecture Implementation

- **Objective:** To implement and train a Tiny Recursive Model for nonogram solving.
- **Tasks:**
    - Implement 2-layer TRM architecture with three information streams (puzzle clues, current solution, latent reasoning state).
    - Design puzzle encoding scheme to represent row/column clues and grid state as embeddings.
    - Implement recursive refinement loop with configurable iteration count (K=16).
    - Apply deep supervision at multiple checkpoints during training and use data augmentation (rotations, reflections) to increase effective training samples.

3. Comparative Evaluation & Analysis

- **Objective:** To systematically compare TRM and classical approaches across multiple dimensions.
- **Tasks:**
    - Perform accuracy comparison: solve rate on test set by puzzle size and difficulty.
    - Conduct efficiency analysis: training cost (GPU hours, samples needed) vs. classical solver development time.
    - Analyze speed comparison: inference time per puzzle for trained TRM vs. classical solver.
    - Execute failure mode analysis to categorize unsolved puzzles and identify systematic weaknesses.

Paper List

Tiny Recursive Models:

- Jolicoeur-Martineau, A. (2025). "Less is More: Recursive Reasoning with Tiny Networks." *arXiv:2510.04871*.
- Wang, G., et al. (2025). "Hierarchical Reasoning Model." *arXiv:2506.21734*.

Classical Nonogram Solvers:

- Wolter, J. (2009). "Survey of Paint-by-Number Puzzle Solvers." *webpbn.com/survey*.
- Tamura, N., et al. (2014). "Constraint Programming Approaches to Nonogram Solving".

Neural Approaches to Logic Puzzles:

- Recent work (2025): "Solving nonograms using Neural Networks." *arXiv:2501.05882*.
- LogicPuzzleRL: "Cultivating Robust Mathematical Reasoning in LLMs via Reinforcement Learning." *arXiv:2506.04821*.

Constraint Satisfaction:

- Rossi, F., et al. (2006). *Handbook of Constraint Programming*. Elsevier.
- Russell, S., & Norvig, P. (2020). "Artificial Intelligence: A Modern Approach" (Constraint Satisfaction chapter).