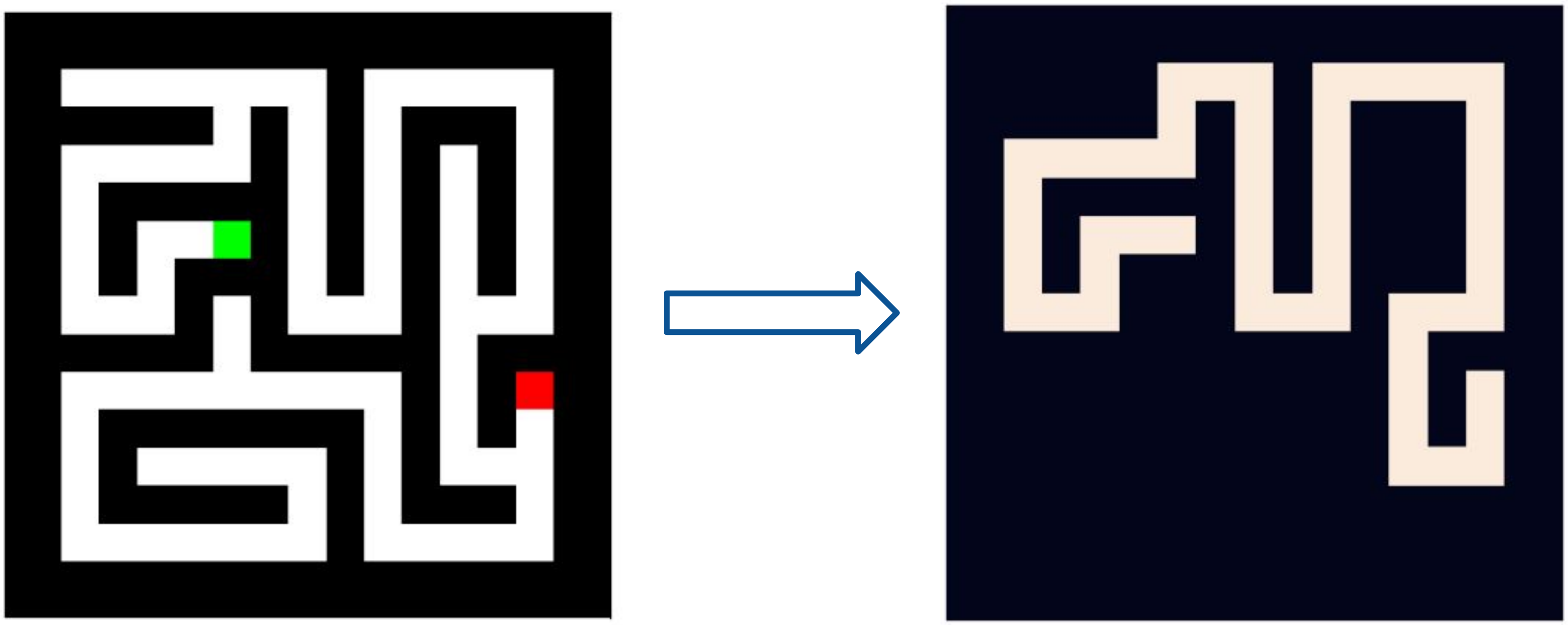
End-to-end Algorithm Synthesis with Recurrent Neural Networks

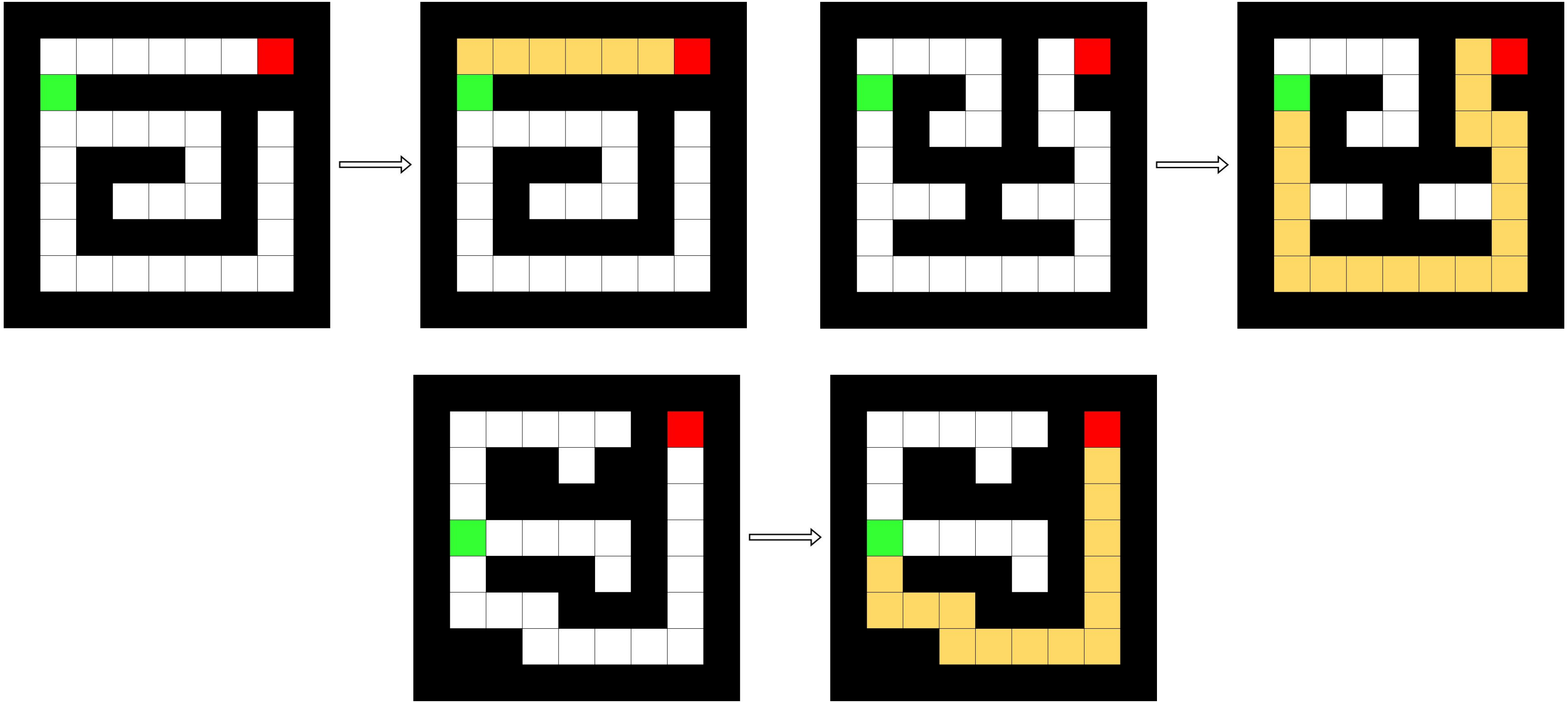
Arpit Bansal, Avi Schwarzschild, Eitan Borgnia,
Zeyad Emam, Furong Huang, Micah Goldblum,
Tom Goldstein

Presentation:
Max Krähenmann

Algorithm Synthesis

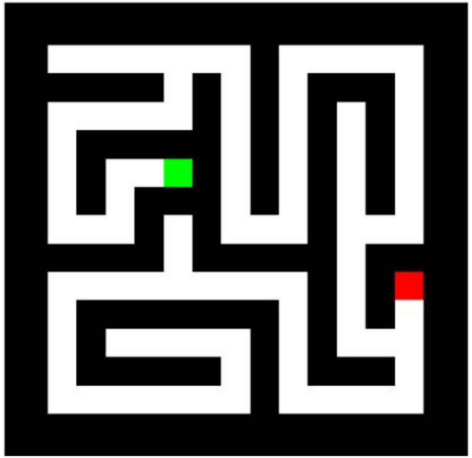


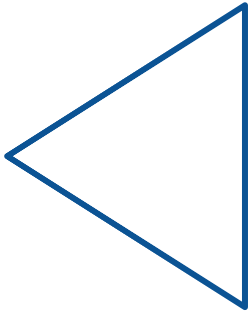
Simple primitives, complex strategies



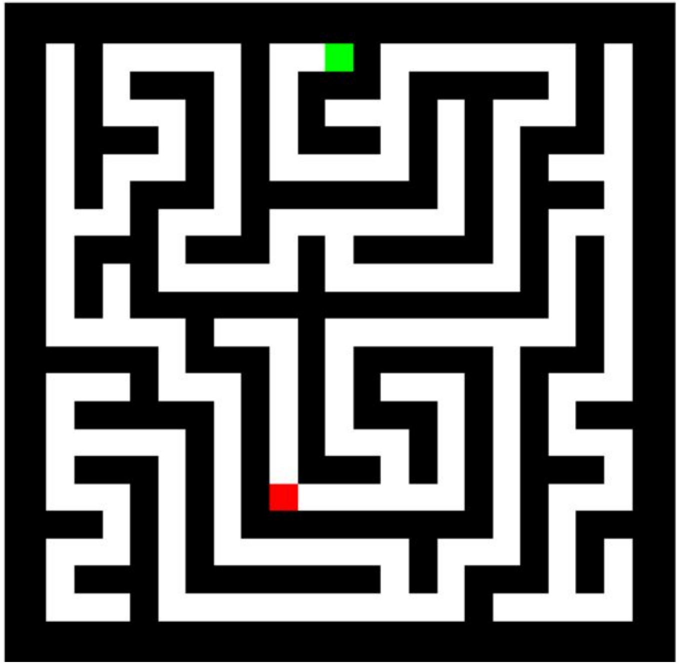
Logical Extrapolation

Learn on this...





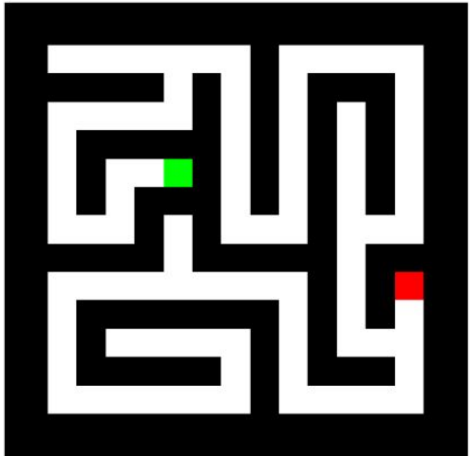
... to be able to solve this



Logical Extrapolation

... or this

Learn on this...



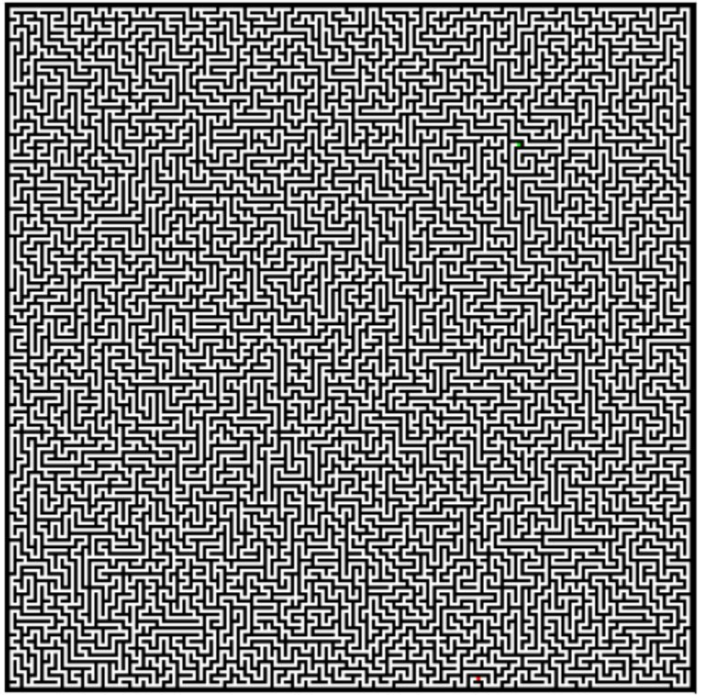
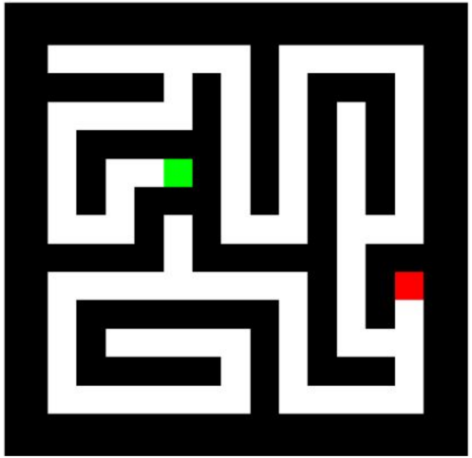
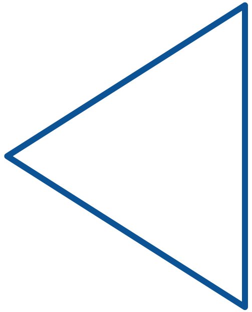


Figure 18: Fun for the whole family!

Logical Extrapolation

Learn on this...





... or this

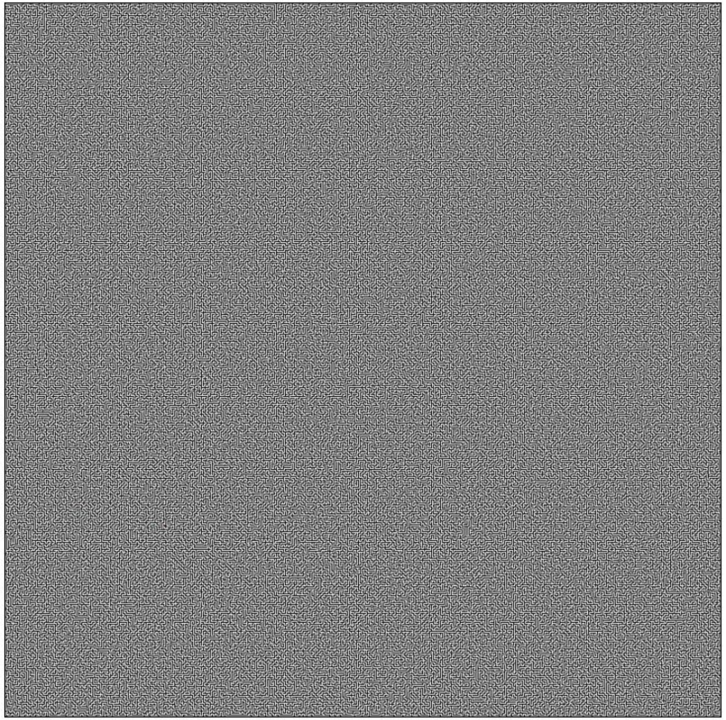
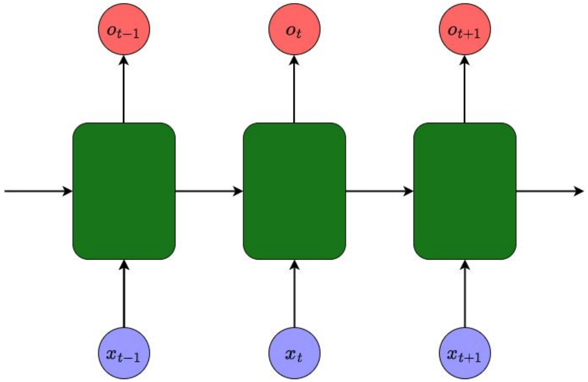


Figure 19: An example of a *very, very* fun 801 × 801 maze.

Related Work

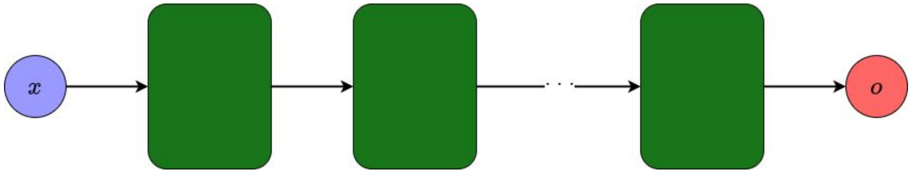
- Classical RNNs
  → amount of computation linked to input size
  → trained to produce one bit at a time



- Hybrid Models
  → not end-to-end

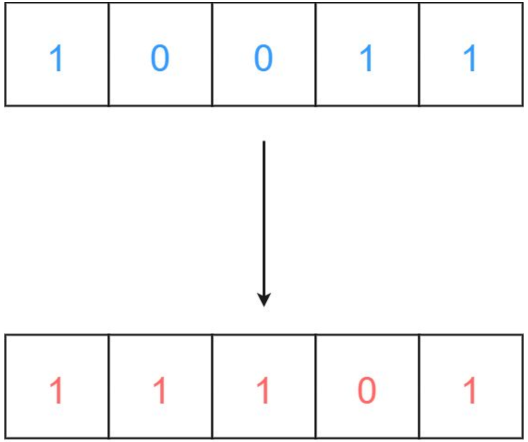
Don’t think harder, think deeper

- Adaptive Neural Nets
  → vary computation based on input
  → all previous work on this was tested in-distribution

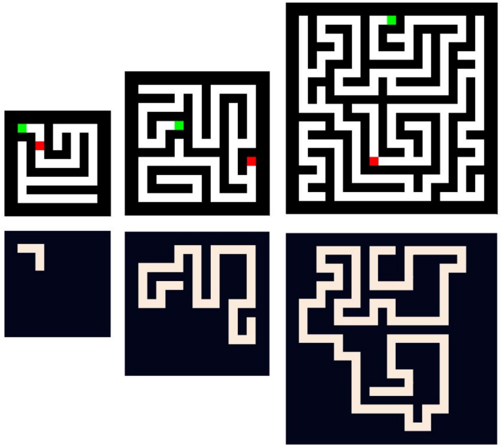


Benchmark Problems

Prefix Sum

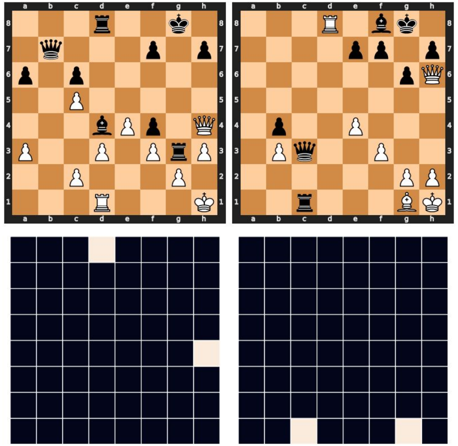


Maze Solving



[1]

Chess Puzzles

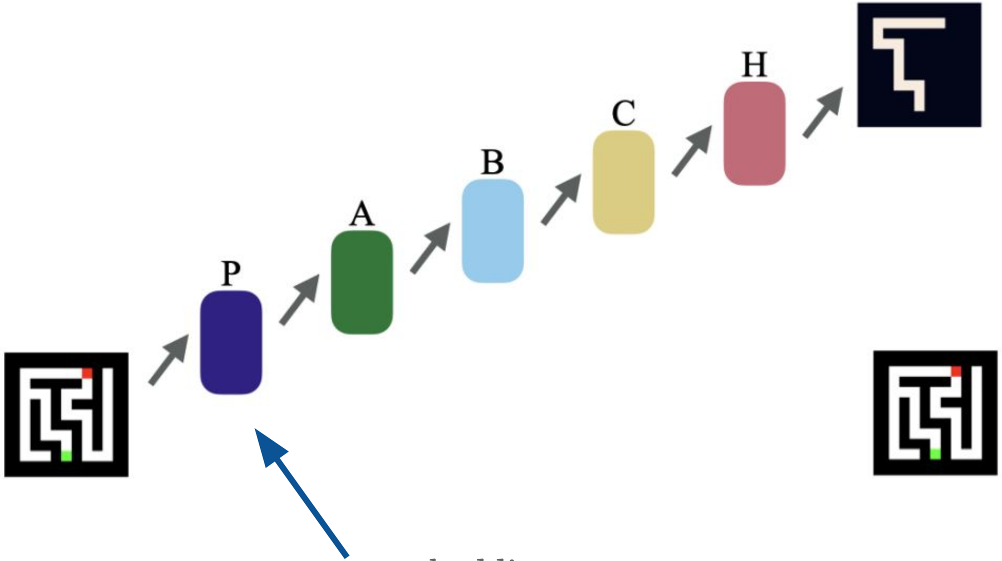


[1]

[1] Schwarzschild et al. *Datasets for studying generalization from easy to hard examples*, 2021

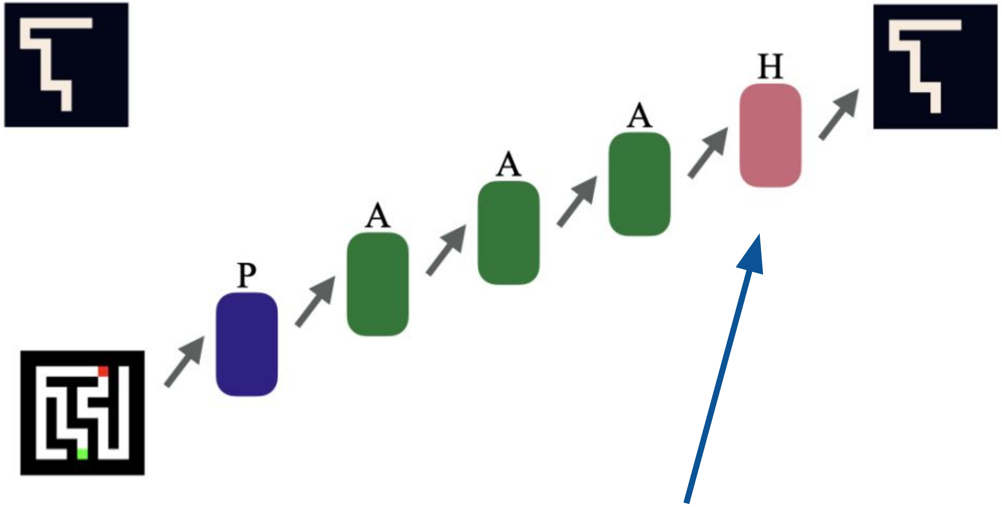
NN Architecture

Feed Forward NN



embedding
function

Deep Thinking (DT) model



map:
features –> output

Problems with extrapolation

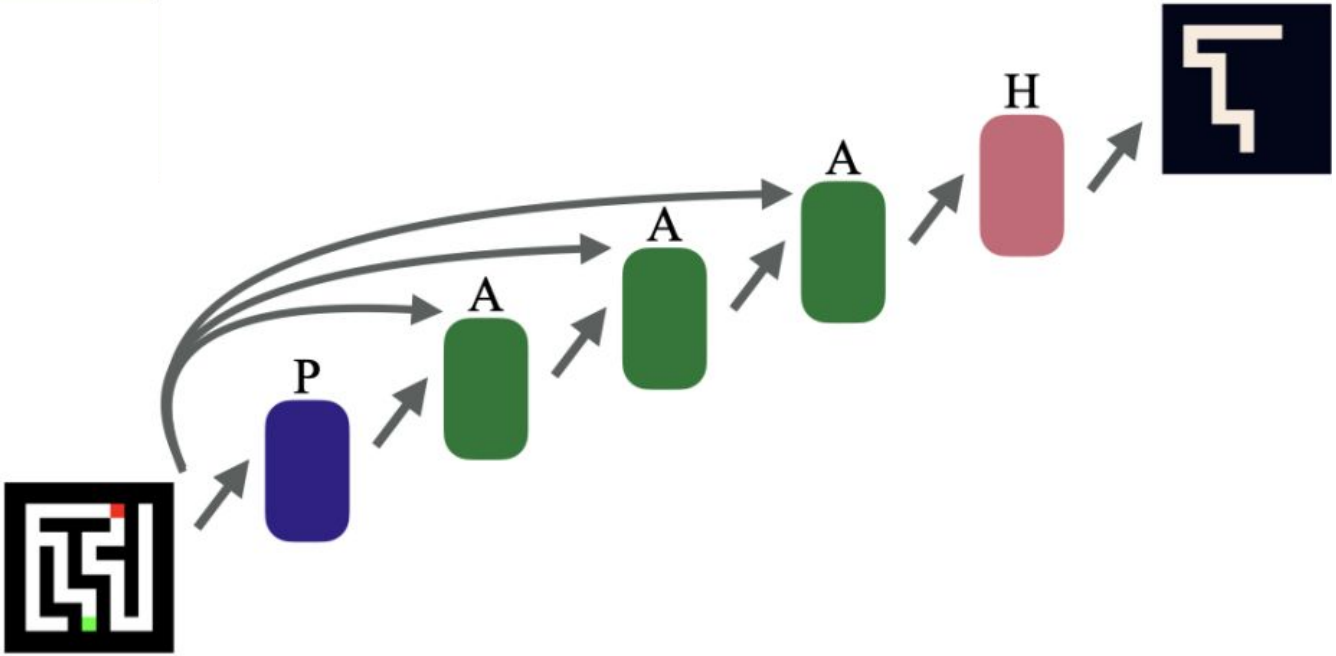
Tested on 48-bit Strings	
Model	Peak Acc. (%)
DT	94.61 ± 1.19
FF	27.15 ± 2.56

Tested on 512-bit Strings	
Model	Peak Acc. (%)
DT	0.00 ± 0.00
FF	0.00 ± 0.00

Tested on 13×13 Mazes	
Model	Peak Acc. (%)
DT	85.59 ± 2.81
FF	38.22 ± 5.28

Tested on 59×59 Mazes	
Model	Peak Acc. (%)
DT	0.00 ± 0.00
FF	0.00 ± 0.00

Recall

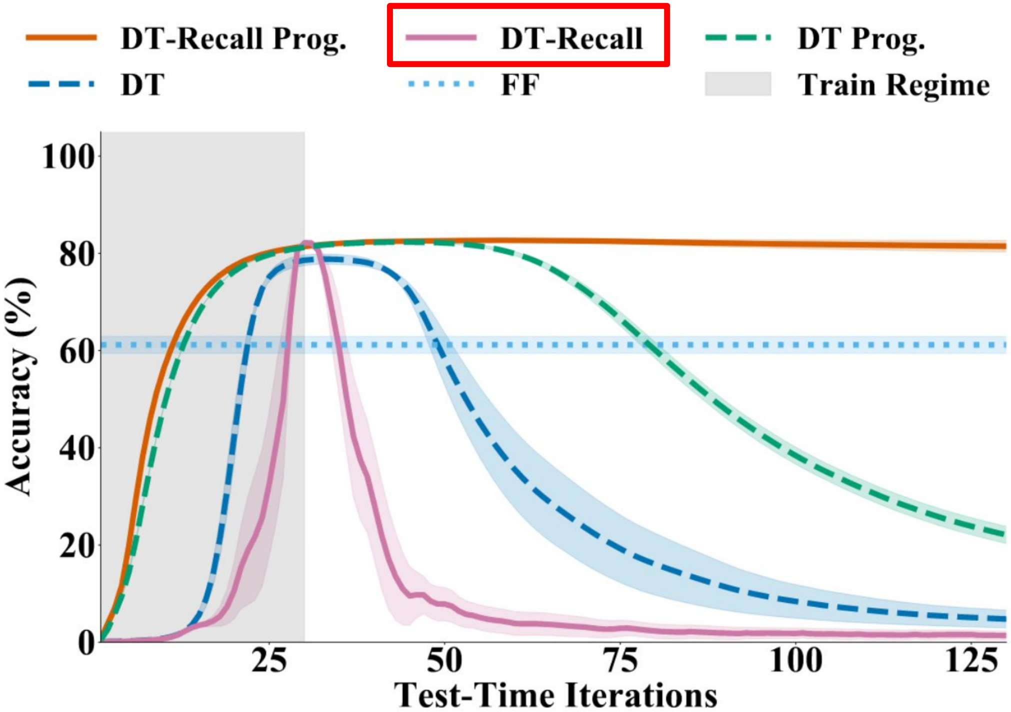


Improvements

Tested on 512-bit Strings	
Model	Peak Acc. (%)
DT	0.00 ± 0.00
DT-Recall	96.19 ± 3.73
FF	0.00 ± 0.00

Tested on 59×59 Mazes	
Model	Peak Acc. (%)
DT	0.00 ± 0.00
DT-Recall	82.72 ± 15.14
FF	0.00 ± 0.00

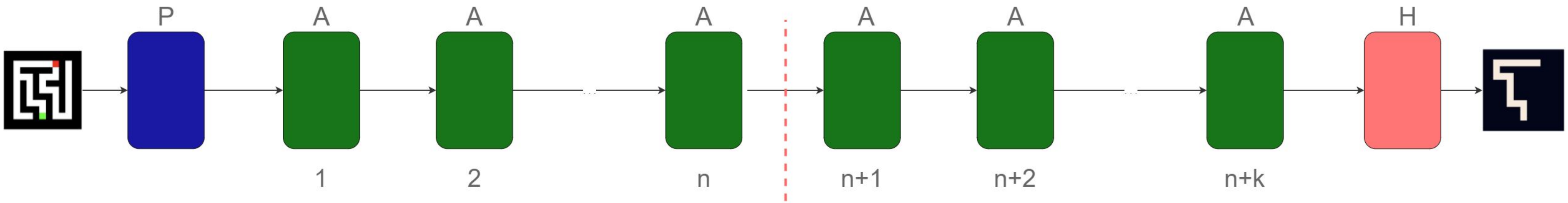
“Overthinking”



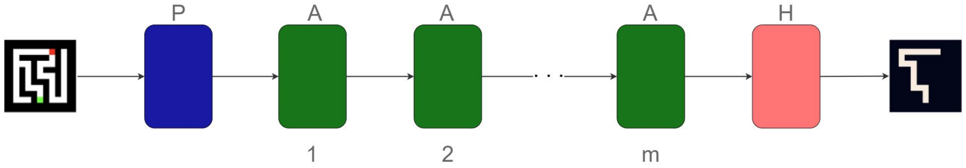
Training with progressive loss

$m \leftarrow$ max. num. of iterations

$n \leftarrow U(0, m-1)$ $\quad$ $k \leftarrow U(1, m-n)$



no gradients $\quad$ $\mathcal{L}_{progressive}$



$\mathcal{L}_{max\_iters}$

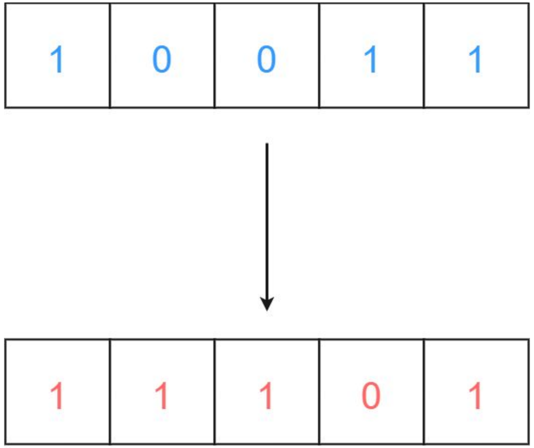
Loss Function

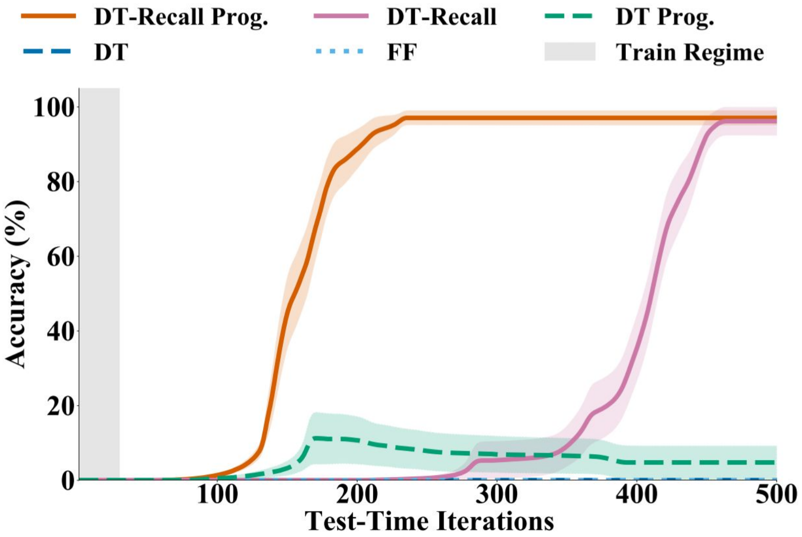
$$\mathcal{L} = (1 - \alpha) \cdot \mathcal{L}_{\text{max\_iters}} + \alpha \cdot \mathcal{L}_{\text{progressive}}$$

Compute $\nabla_\theta \mathcal{L}$ and update θ

Results: Prefix Sum

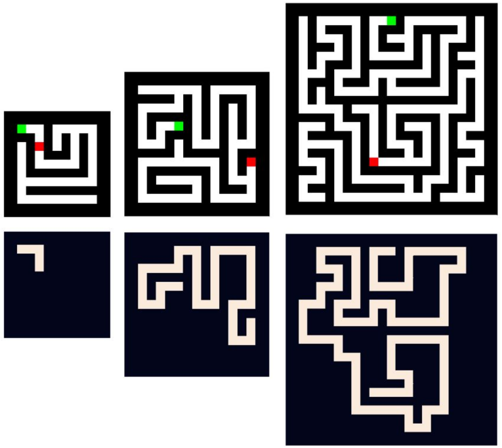
- Trained on 32-bit data and evaluated on 512-bit data

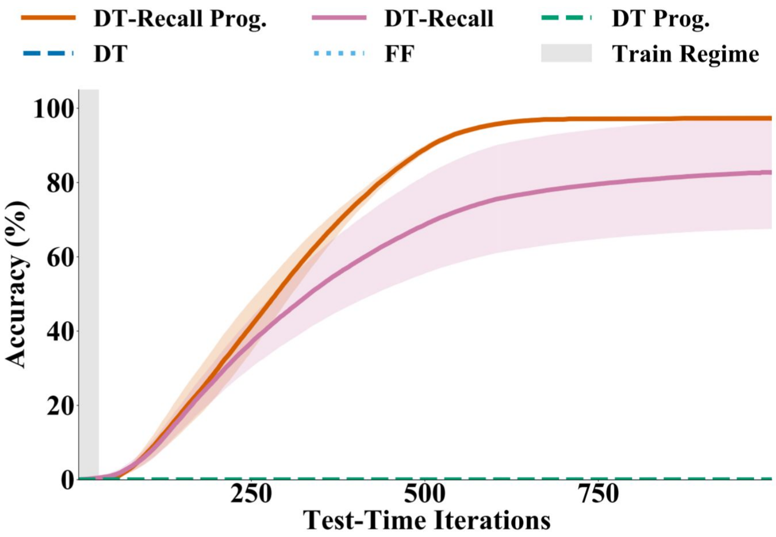




Results: Maze Solving

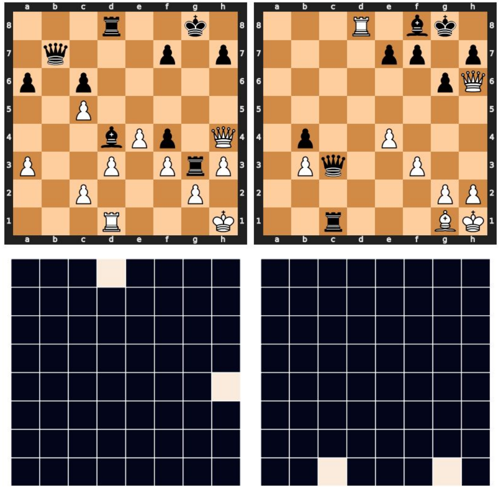
- Trained on 9x9, evaluated on 59x59

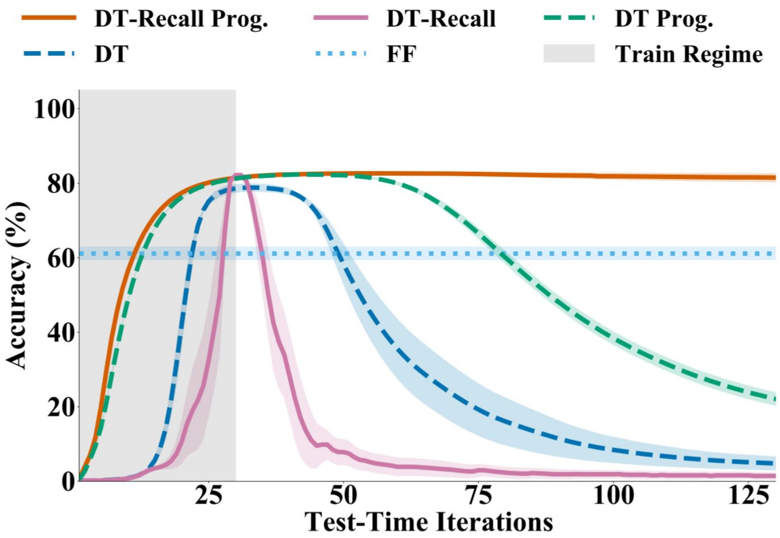




Results: Chess Puzzles

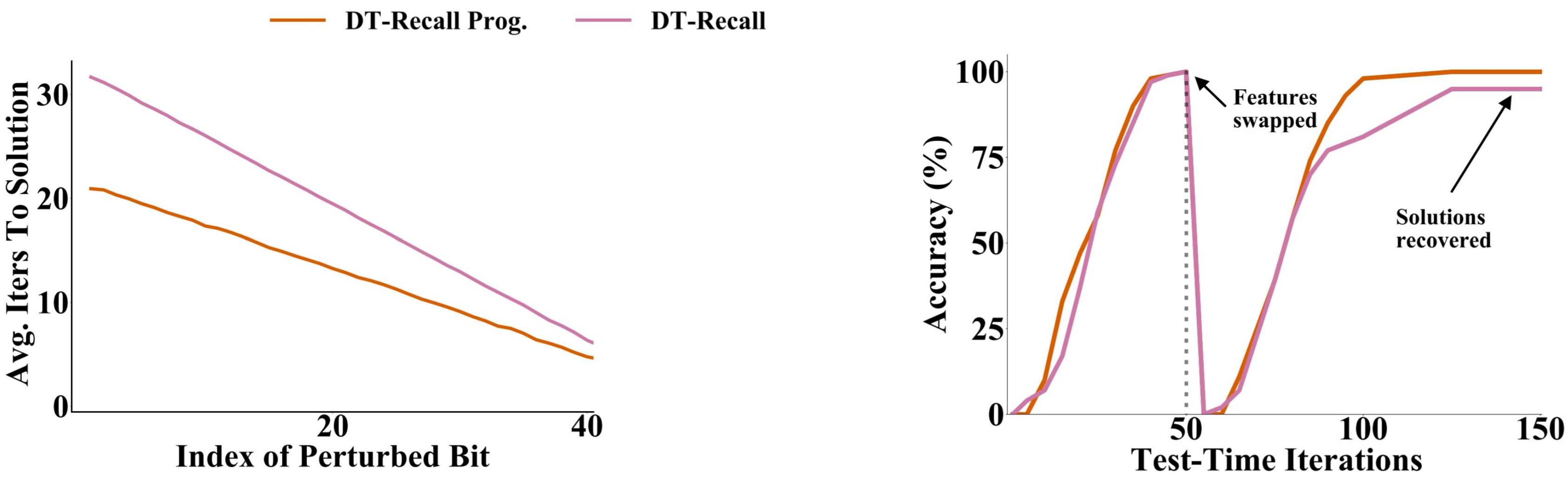
- Trained on 600k easiest, evaluated on 600k-700k easiest





Manipulating Inputs

- How long to recover
- What happens when features are swapped



What does the Model really do?

Problem:

1 0 1 0 1 0 0 0 1 0 1 1 0 1 0 0 1 1 0 0 1 1 1 1 0 1 0 1 0 0 0 0

Target:

1 1 0 0 1 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 0 1 0 1 1 0 0 1 1 1 1 1

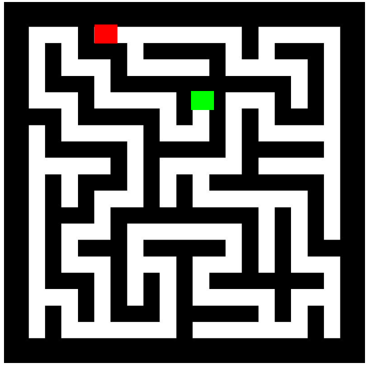
Iterations:

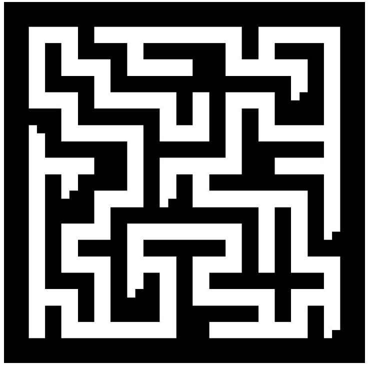
1 1 0 0 1 1 1 0 1 0 0 1 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 1 1 1 0 1
1 1 0 0 1 1 1 1 0 0 0 1 1 0 0 0 1 0 0 1 0 1 0 1 1 0 0 1 1 1 0 1
1 1 0 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 1 1 1 1
1 1 0 0 1 1 1 1 0 0 1 0 0 1 1 1 1 0 0 0 1 0 1 0 0 1 0 1 1 1 1 1
1 1 0 0 1 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 1 0 1 0 0 1 1 0 0 1 1 1
1 1 0 0 1 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 0
1 1 0 0 1 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1
1 1 0 0 1 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 0 1 0 1 1 0 0 1 1 1 1 1

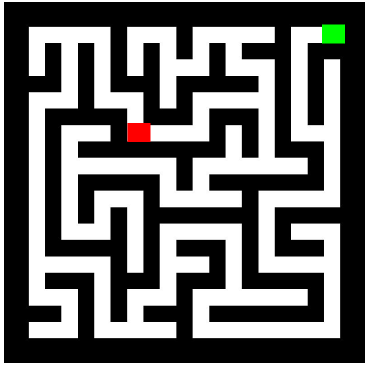
What does the Model really do?

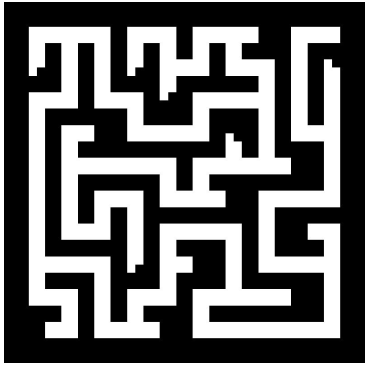
1 0 1 0 1 0 1 0 1 0 0 1 0 0 1 1 0 1 1 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 0 1 0 0 0 1 1 0 0 0 1 1 1 1 0 1
1 0 1 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 1 1 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 0 0 1 1
1 0 1 0 1 0 0 1 1 1 1 1 0 1 0 1 0 1 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 1 1 0 1 0 1 1 1 0 0 0 0 1 1
1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 1 0 0
1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 0
1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 1 1 0 1 1 1 0 1 0 0 0 1 1 0 0 1 1
1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 0 1 1
1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 0 1 1
1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 0 0 1 1
1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0 1 1
1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 1 0 0 0 1 0 1 1 0 1 1 0 0 1 1
1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 1 0 0 0 1 0 1 1 1 0 0 1 0 1 1
1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 1 0 0 0 1 0 1 1 1 0 0 1 1 0 0

What does the Model really do?

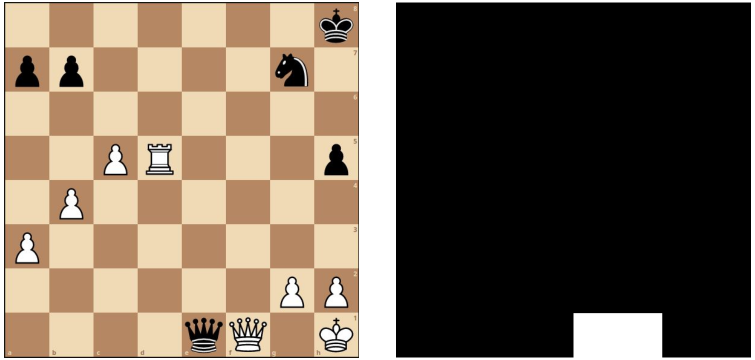


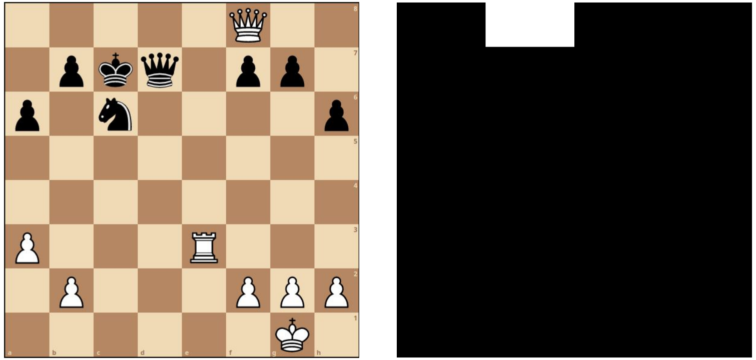


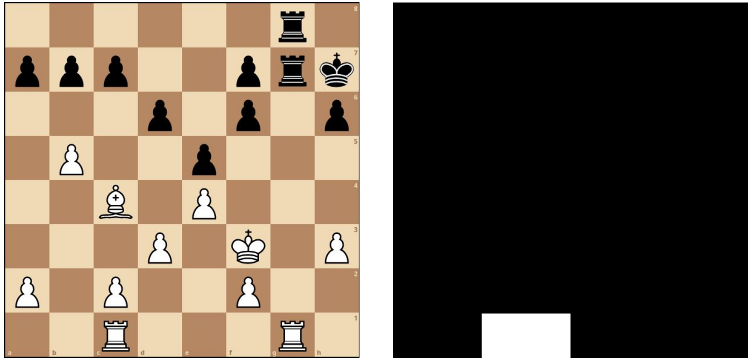




What does the Model really do?

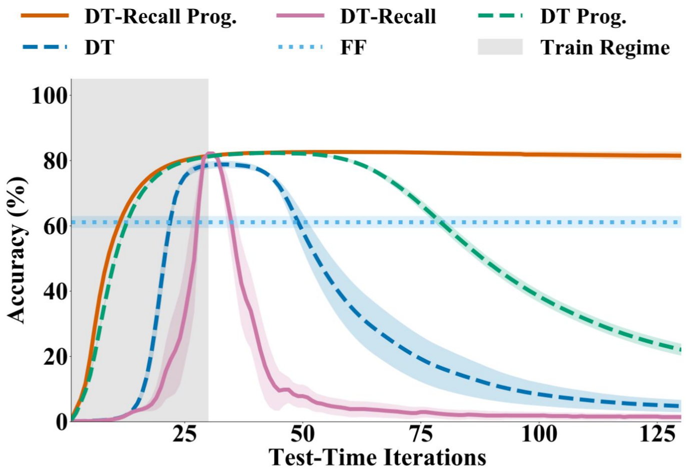




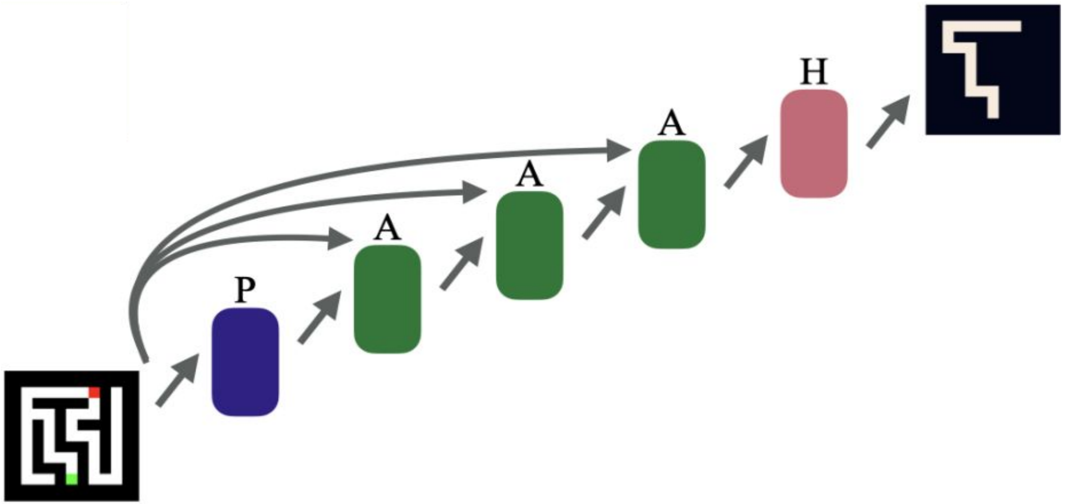


Conclusion: Problems

Tested on 512-bit Strings	
Model	Peak Acc. (%)
DT	0.00 ± 0.00
DT-Recall	96.19 ± 3.73
FF	0.00 ± 0.00

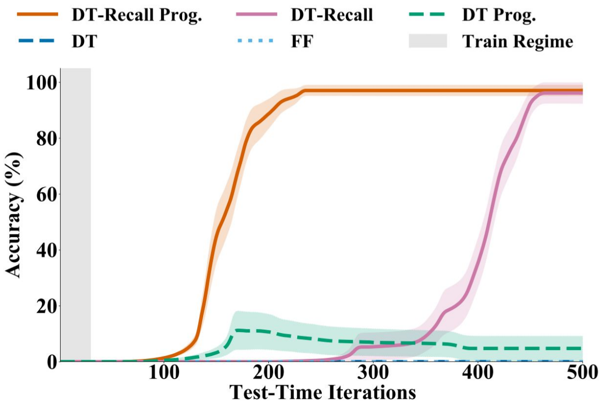


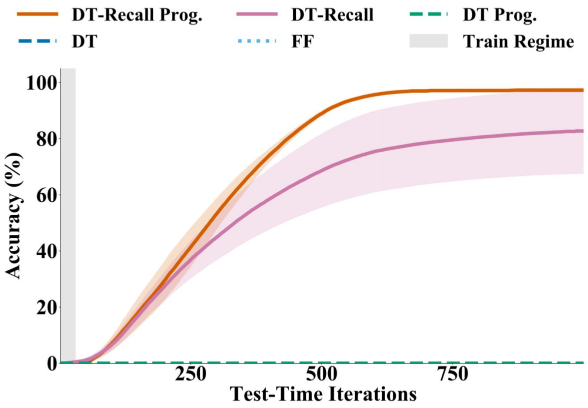
Conclusion: Main Contributions

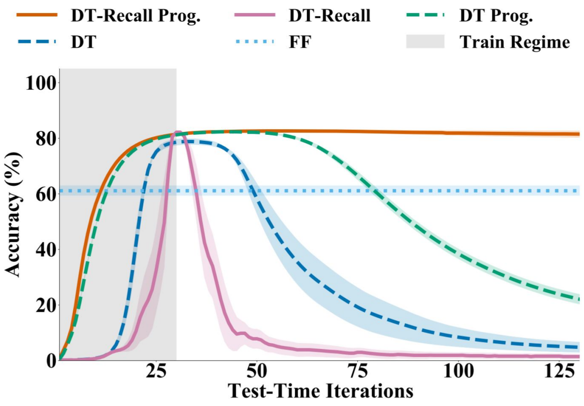


$$\mathcal{L} = (1 - \alpha) \cdot \mathcal{L}_{\text{max\_iters}} + \alpha \cdot \mathcal{L}_{\text{progressive}}$$

Conclusion: Results







Discussion: strong points

- Ideas simple and useful
- Extrapolation to bigger problems not often done
- Good contribution to algorithm synthesis

Discussion: weak points

- More baselines needed to properly assess performance

- Benchmark problems are toy-ish

- No investigation as to what the Net is doing in its “thinking” process