WHEN INTERMEDIATE SUPERVISION DOESN'T HELP: EVIDENCE FROM RECURRENT CNNS

Elisa Klunder*
University of Groningen
Groningen, The Netherlands
e.klunder.1@student.rug.nl

Steven Abreu, Guillaume Pourcel
University of Groningen
Groningen, The Netherlands
{s.abreu,g.a.pourcel}@rug.nl

ABSTRACT

Intermediate supervision has shown promise for mathematical reasoning, logical inference, and algorithmic tasks, yet recent evidence questions whether these improvements reflect genuine algorithmic reasoning. We test this in maze-solving, teaching recurrent convolutional neural networks systematic search strategies inspired by classical computer science algorithms. Networks successfully replicate the search strategies at test time, yet intermediate supervision consistently underperforms end-to-end learning across all generalisation tests, and performance collapses across all search strategies when networks are tested on mazes with different topologies. These findings suggest that explicit procedural guidance fails to teach networks transferable algorithmic principles. By forcing interpretable traces, such explicit thinking may even constrain network flexibility, challenging assumptions about intermediate supervision's benefits for algorithmic reasoning.

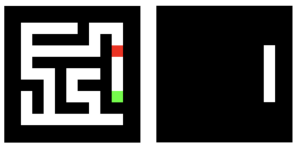
1 INTRODUCTION

The past decade has witnessed a shift in how neural networks approach complex reasoning tasks, with intermediate supervision techniques achieving impressive performance gains across mathematical reasoning, logical inference, and complex question-answering domains (Wei et al., 2022; Zelikman et al., 2022; Creswell et al., 2022; Lightman et al., 2023). These successes have led to the adoption of process (also called "step-by-step" or "intermediate") supervision approaches, based on the assumption that intermediate tokens represent semantically meaningful reasoning steps that guide the model toward correct final answers.

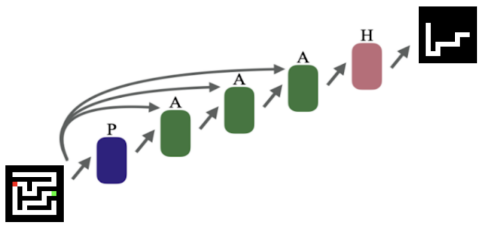
However, more recent research streams have begun questioning the source of process supervision's benefits. Specifically, do these benefits really stem from genuine logical extrapolation—the ability to generalise from simple to complex tasks outside the training distribution through extended reasoning? This capacity is also sometimes called "upwards generalisation" or "algorithmic reasoning" (Veličković et al., 2022). Some evidence has suggested that the superior performance achieved by process-supervised models may result from computational mechanisms that do not rely on the semantic content of reasoning traces (Stechly et al., 2025), challenging the assumption that process supervision teaches networks robust and interpretable reasoning capabilities.

The present work addresses this question through an investigation of how intermediate supervision affects neural network learning in maze-solving tasks, a controlled experimental domain where correct "reasoning" procedures may be precisely specified. We extend the methodological foundations of Schwarzschild et al. (2021) and Bansal et al. (2022), who presented "thinking" recurrent networks that dynamically adjust their computational depth during test-time. Their approach demonstrated that networks trained on simple maze navigation tasks with limited iteration budgets could solve substantially larger mazes by "thinking longer" during inference. Building on their work, we teach recurrent convolutional neural networks (recurrent CNNs) multiple distinct maze-solving strategies through step-by-step guidance and evaluate whether such supervision produces robust logical extrapolation capabilities.

*Corresponding author



(a) Example input maze and target, start position in red and goal position in green.



(b) DT-Network with recall from Bansal et al. (2022).

Figure 1: *Left*: example of input maze and target, *right*: DT-Network architecture.

The solution is evaluated along three dimensions. First, we test whether networks can learn to replicate explicit search strategies inspired by traditional maze-solving algorithms that reliably extrapolate to harder instances. Second, we assess whether this intermediate supervision provides generalisation benefits over end-to-end training by evaluating performance on progressively larger mazes. Third, we examine the extrapolation capabilities of learned strategies under domain shifts, including changes in goal positions and percolation degree, which test extrapolation beyond the training distribution. Our contributions can be summarised as follows:

- We provide evidence that intermediate supervision in maze-solving tasks fails to teach robust logical extrapolation, producing brittle behaviours that collapse under domain shifts, thus transferring the findings of Stechly et al. (2025) to a new training setup.
- We support the notion that forcing networks to adopt fixed reasoning strategies through process supervision might limit their ability to discover more effective problem-solving approaches, challenging optimistic interpretations of intermediate supervision effectiveness.
- We contribute to the evidence that current intermediate supervision approaches may overestimate their effectiveness in teaching genuine reasoning capabilities, informing the development of more reliable approaches to neural reasoning systems.

These findings suggest that alternative approaches may be necessary for developing robust reasoning capabilities that generalise beyond training distributions and maintain performance under domain shifts. Additionally, our results suggest that implicit reasoning, unconstrained by intermediate supervision, generalises more robustly than explicit, human-interpretable reasoning traces.

1.1 Learning to solve mazes with intermediate supervision

This study focuses on testing process supervision in maze-solving as a controlled experimental domain where intermediate reasoning procedures can be precisely specified. As shown in Figure 1a, mazes are represented as RGB pixel images $M \in \mathbb{R}^{H \times W \times 3}$ with start position $s \in \mathbb{N}^2$ (in red) and goal position $g \in \mathbb{N}^2$ (in green), and the learning objective of finding a valid path from s to g that avoids wall cells.

The networks must learn to navigate through sequential decision-making processes. Unlike previous research that focuses primarily on end-to-end learning, the approach explicitly provides intermediate supervision signals that specify the desired reasoning strategy at each step. Networks receive a sequence of binary masks that represent the steps of a specific search strategy for solving the maze, guiding the networks toward an iterative problem-solving approach. We initially expected that supervising the network with algorithmically correct strategies would improve generalisation, as these strategies should, in principle, apply across all mazes.

A central assumption of our method is that meaningful reasoning strategies should exhibit systematic patterns that generalise across problem instances, maze topologies, and sizes. Networks that exhibit true logical extrapolation should demonstrate consistent performance when confronted with domain shifts such as changes in the goal position, changes in the likelihood of a maze containing loops (also called "percolation"), and scaling to larger problem instances (mazes with higher H and W values). An example of variations in goal position and percolation value can be seen in Figure 3.

2 METHODS

2.1 MODEL ARCHITECTURE

We build on the recurrent CNN architecture of Bansal et al. (2022), which iterates a weight-shared residual block for T steps over an RGB maze image $M \in \mathbb{R}^{H\times W\times 3}$, producing per-pixel path predictions $\mathbf{y} \in \{0,1\}^{H\times W}$. An encoder P_{θ_P} projects the input into a feature space, a recurrent block A_{θ_A} refines representations iteratively with a recall connection to the original input, and a decoder H_{θ_H} produces two-channel output logits (Figure 1b). This architecture allows dynamic computational depth at test time: networks trained with a fixed iteration budget can allocate more iterations for harder problems. Full architectural details are provided in Appendix A.1.

2.2 INTERMEDIATE SUPERVISION FRAMEWORK

The main contribution of this work lies in the systematic generation of intermediate supervision signals that guide networks toward specific algorithmic reasoning strategies. Unlike previous approaches that use only final outcome supervision, this framework guides the computation at each step.

Architecturally, we realise this by extending the recurrent module to directly condition on, and be trained to refine, its own intermediate predictions. The recurrent block receives the previous output logits as additional input channels, operating on $\mathbb{R}^{H\times W\times(D+3+2)} \to \mathbb{R}^{H\times W\times D}$ instead of $\mathbb{R}^{H\times W\times(D+3)} \to \mathbb{R}^{H\times W\times D}$, concatenating the current internal state, the original maze image, and the most recent two-channel output logits. This output-space feedback allows the network to condition each iteration on its own past predictions, while the supervision signals themselves are generated externally. The forward pass with output-space feedback becomes:

$$\mathbf{h}_t = A_{\theta_A}(\mathbf{h}_{t-1}, M, \mathbf{O}_{t-1})$$
$$\mathbf{O}_t = H_{\theta_H}(\mathbf{h}_t)$$

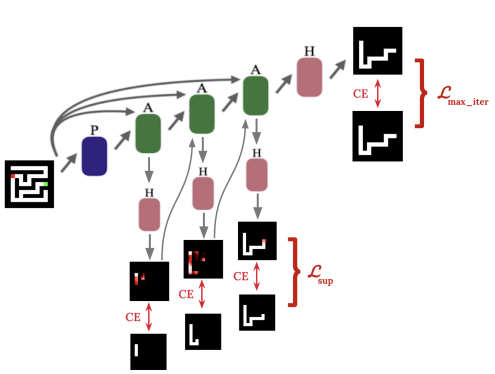
where $\mathbf{O}_{t-1}$ represents the output logits from the previous iteration.

While the final predictions are binary masks, the intermediate states are best understood as continuous confidence maps. For visualisation purposes (e.g., in Figure 4), when plotting intermediate outputs these logits are converted to probabilities via a Softmax function and displayed using a colourmap where black represents the background ($p \approx 0$), white represents the path ($p \approx 1$), and red tones indicate intermediate confidence levels ($0 < p < 1$).

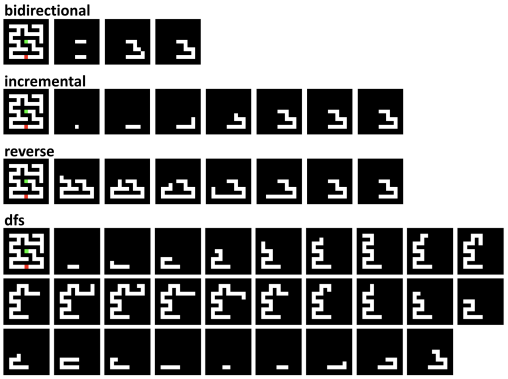
The supervision signals are generated by an intermediate-supervision generator that implements multiple pathfinding strategies adapted from classical graph algorithms, producing visually interpretable reasoning traces. Each strategy produces a sequence of binary masks $\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_L$ where $\mathbf{M}_i \in \{0,1\}^{H\times W}$ indicates which pixels should be activated at iteration i according to the chosen strategy. The total number of supervision masks L depends on the maze size, the selected search strategy, and a *step* parameter, which controls the temporal resolution of the supervision. Specifically, the *step* parameter determines how many intermediate masks are skipped between supervised steps during training (i.e., a step size of k uses every k-th mask in the full sequence). This allows training to focus on broader transitions in the search process, reducing redundancy and computation cost.

The search strategies are described below. The strategies adopt classical algorithms to be visually interpretable. This produces traces that follow iterative and generalisable search strategies that can be understood visually, without requiring accessory memory or score-calculation mechanisms.

- The *incremental search* strategy progressively reveals the optimal path from start to goal, mimicking a forward search process that reaches the solution step by step. This is implemented by incrementally growing the shortest path obtained from breadth-first search.
- The *bidirectional search* strategy simultaneously reveals path segments from both endpoints, teaching networks to search for a solution from both endpoints.
- The *reverse exploration* strategy begins with the complete exploration state produced by breadth-first search and iteratively removes dead ends until only the optimal path remains, teaching networks to prune irrelevant search branches.



(a) Illustration of the training protocol for one training sample with intermediate supervision signal produced by solver.



(b) Intermediate-supervision masks produced for the same 9x9 maze by different search strategies.

- The *depth-first search* strategy implements classical depth-first exploration with explicit backtracking visualisation, teaching networks to handle systematic exploration with recovery from dead ends.

An example of the supervision traces produced by each search strategy for a 9x9 maze can be seen in Figure 2b. During training, at the end of each supervision trace, 10 masks with the correct shortest path are appended to aid with the stable convergence of solutions.

The training objective uses cross-entropy loss applied to pixel-wise binary classification. Each training instance provides binary target labels for every spatial location, with the loss averaged across all output pixels.

The new training framework combines end-to-end learning with intermediate supervision. The total loss function balances final accuracy with adherence to prescribed reasoning processes through:

$$\mathcal{L}_{\text{total}} = (1 - \alpha) \cdot \mathcal{L}_{\text{max\_iter}} + \alpha \cdot \mathcal{L}_{\text{sup}} \tag{1}$$

where $\mathcal{L}_{\text{max\_iter}} = \text{CE}(\mathbf{O}_{T_{\text{max}}}, \mathbf{y}_{\text{target}})$ is the pixel-wise cross-entropy loss computed between the network's final-iteration logits and the ground-truth target mask, maintaining pressure for correct final outcomes, while $\mathcal{L}_{\text{sup}}$ enforces adherence to intermediate reasoning steps.

The intermediate-supervision loss handles variable-length strategy sequences in the same training batch through normalisation across both spatial dimensions and temporal steps:

$$\mathcal{L}_{\text{sup}} = \frac{1}{B} \sum_{b=1}^{B} \frac{\sum_{t=1}^{T_{\text{max}}} \text{CE}(\mathbf{O}_{b,t}, \mathbf{M}_{b,t}) \cdot \mathbb{I}_{\text{valid}}(b,t)}{\sum_{t=1}^{T_{\text{max}}} \mathbb{I}_{\text{valid}}(b,t)} \tag{2}$$

where $\mathbf{O}_{b,t}$ is the network output for sample b at iteration t, $\mathbf{M}_{b,t}$ is the corresponding ground-truth supervision mask, and B is the batch size. The indicator $\mathbb{I}_{\text{valid}}(b,t)$ indicates whether step t exists for sample b, which handles varying sequence lengths across batch elements. The loss is computed by first normalizing per sample, then averaging across the batch. T_{max} is set to the maximum supervision length within the current batch.

The hyperparameter $\alpha \in [0, 1]$ controls the balance between final accuracy and intermediate supervision, with $\alpha = 0$ reducing to traditional end-to-end training and $\alpha = 1$ providing pure step-by-step supervision. Complete details of all hyperparameters and other training details are provided in Appendix A. [1]

A simplified overview of the new training procedure for one sample can be seen in Figure 2a.

[1] Code implementation is available at https://github.com/elisaklunder/intermediate-supervision-RNNs

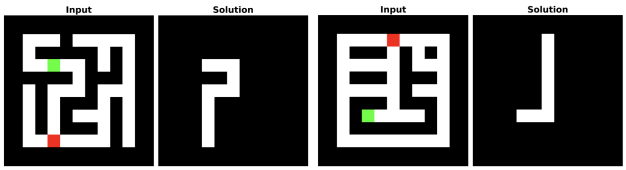


Figure 3: Maze input-solution pairs of size 9×9. Example of maze without loops ($p = 0$) and with goal not in a dead-end (left), and example of a percolated maze ($p = 0.2$) with loops and goal in a dead-end (right).

2.3 Stable Testing Protocol

This work also introduces a stable testing protocol, inspired by Adaptive Compute Time (Graves, 2016), that allows networks to run until their predictions stabilise. The stability criterion combines output consistency with confidence thresholds. A prediction is stable when both:

- the network produces identical binary masks for $n = 5$ consecutive iterations after softmax thresholding at 0.5 (on the path class), indicating convergence to a consistent prediction,
- and all predicted path pixels on traversable (non-wall) cells exhibit confidence scores exceeding $p_{\text{thresh}} = 0.9$, ensuring high certainty in the final solution.

The protocol maintains computational efficiency through early termination when stability criteria are met, while imposing a maximum iteration limit of $T_{\max} = 1000$ to prevent infinite computation in cases where networks fail to converge. More details about how this value was chosen are available in Appendix A.2

We evaluate networks using:

- *Stable accuracy*, which measures final correctness using stabilised predictions, and constitutes the primary indicator of reasoning effectiveness.
- *Mean iterations* per search strategy and maze size were tracked to understand average computational requirements across test instances.

2.4 Datasets

We primarily use the Easy-to-Hard dataset by Schwarzschild et al. (2021), which provides systematic variation in maze size. This dataset features tree-structured mazes without cycles, where goal positions consistently appear at dead ends, creating mazes that vary only in size. The dataset comprises 50,000 training instances and 10,000 testing instances across maze sizes $9, 11, 13, 15, 17$, with additional testing sets of 1,000 instances each for larger sizes $19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 59$. The tree structure constraint simplifies the reasoning task by ensuring unique paths between any two locations, eliminating ambiguity about optimal routes. Goal placement at dead ends provides natural termination points, creating clear success criteria for algorithmic reasoning.

In addition, our approach employs the Maze-Dataset package (Ivanitskiy et al., 2023) as used by Knutson et al. (2025) to test reasoning robustness beyond size scaling. This dataset includes configurable maze generation parameters that modify different maze properties:

- the percolation parameter $p \in [0, 1]$ controls loop density and allows testing whether reasoning strategies remain effective when unique paths are not guaranteed,
- the dead-end goal parameter $d \in (True, False)$ controls whether the goal can be placed in a dead-end. By default ($d = True$), the goal can appear in dead-end states; setting $d = False$ enforces the harder condition that goals are never located in dead-ends.

Figure 3 shows the difference between an example maze from the Easy-to-Hard package (left) and one from the Maze-Dataset package (right).

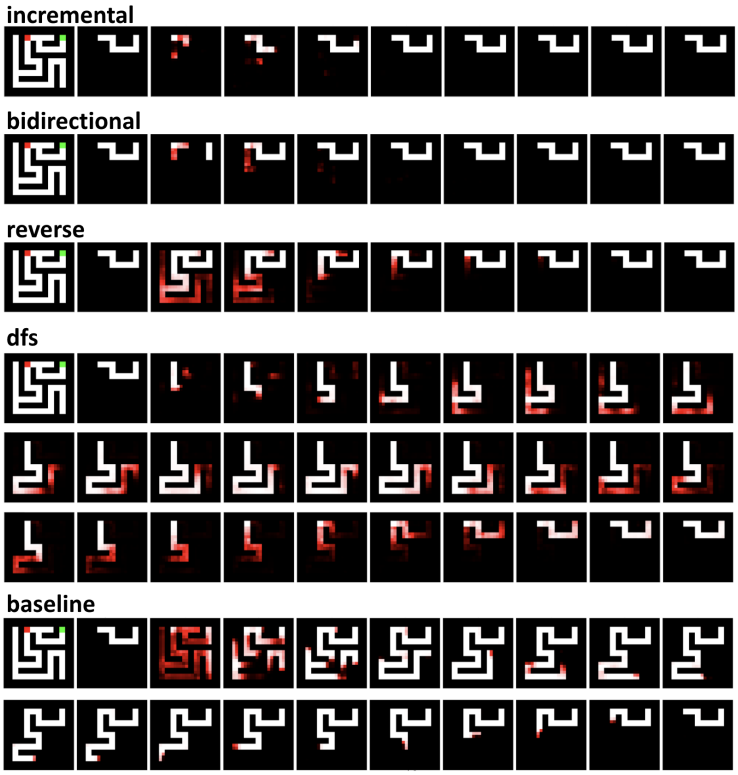


Figure 4: Example of learned reasoning strategies on the same 9x9 maze. The first two images for every strategy are the input and target. In the intermediate steps, black indicates pixels predicted to be outside the shortest path, white indicates high confidence that pixels belong to the shortest path, and red tones indicate intermediate confidence. All intermediately supervised networks reproduce their prescribed search strategies and converge to the correct result, except the baseline, which follows no interpretable algorithm yet solves the maze nonetheless.

3 Results

This section is organised around the three core research questions of this work: whether networks can learn prescribed reasoning strategies, whether intermediate supervision provides generalisation advantages, and whether learned strategies exhibit robustness to domain shifts.

3.1 Strategy Learning

We first examined whether recurrent CNN networks could learn distinct algorithmic reasoning strategies when provided with step-by-step supervision. Networks were trained on 9×9 mazes using four different supervision strategies: incremental search, bidirectional search, reverse exploration, and depth-first search with backtracking.

Visual inspection of the intermediate masks generated by the models at test time shows that networks learned to follow the strategies used for training across all four approaches (Figure 4). Networks trained with incremental supervision demonstrate progressive path construction from start to goal, while those trained with bidirectional supervision exhibit simultaneous exploration from both endpoints. Networks trained on reverse strategy initially produce full exploration maps before iteratively pruning dead-end branches, and networks trained with depth-first search supervision reproduce both forward exploration and backtracking behaviours. Networks trained end-to-end finally exhibit no resemblance to any known algorithm for maze solving in the way they reach the correct solution.

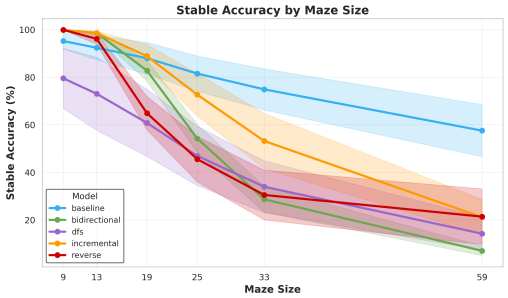
All strategies converged within 50 training epochs, with final training accuracies exceeding 99.9%. All strategies generalised well to unseen mazes of the same size in the test set, achieving test accuracies above 95% in all cases except for the depth-first-search strategy, which achieved around 80% stable accuracy (Table 1).

Table 1: Mean, Standard Deviation (SD), and Standard Error (SE) for model performance on 9x9 test mazes across training runs.

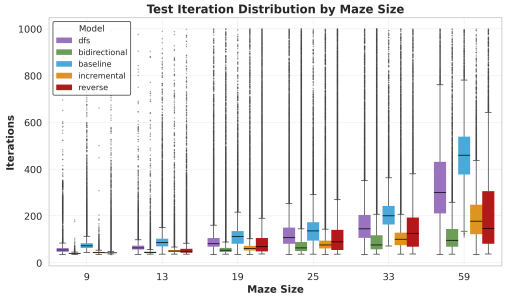
Strategy	Mean	SD	SE
baseline	95.26	10.58	3.346
bidirectional	99.95	0.1140	0.0403
dfs	79.61	35.86	12.68
incremental	99.85	0.3526	0.1246
reverse	99.98	0.0296	0.0098

3.2 Size Generalisation

Second, we tested whether intermediate supervision improves generalisation to larger problem instances compared to end-to-end learning. Networks trained on 9×9 mazes were evaluated on progressively larger test sets ranging from 13×13 to 59×59 mazes using the stable testing protocol.



(a) Stable accuracy by maze size across different training approaches. Intermediate supervision improves scalability but still underperforms end-to-end training at all problem sizes.



(b) Test iteration distribution by maze size and strategy. Box plots show median, quartiles, and outliers for the number of iterations required to reach stable predictions across different supervision approaches.

Although networks trained with intermediate supervision retained non-trivial accuracy on larger maze instances, their performance consistently fell below the end-to-end baseline at every tested size (Figure 5a). Among the supervised strategies, incremental and bidirectional approaches degraded most slowly, staying above 50% accuracy up to 25×25 mazes, but neither matched the baseline for 25×25 mazes or beyond. The complete performance breakdown by model and maze size is provided in Appendix A.5

Computational efficiency also varied by strategy. Bidirectional and incremental networks scaled most efficiently, requiring approximately 131 and 198 iterations on 59×59 mazes, while the baseline showed the steepest growth to 467 iterations (Figure 5b).

3.3 Domain Shift Robustness

Third, we tested strategy robustness under domain shifts by introducing loops (percolation parameter p) into maze structures. As percolation increases, mazes become increasingly connected with fewer dead ends available. Goals were always placed at dead ends during training, but freely positioned during these tests.

The results revealed severe brittleness in all intermediate supervision approaches when confronted with this domain shift. On 9×9 mazes with loops, networks trained with intermediate supervision achieved stable accuracies of 0.1% or lower across all strategies and all training modes (Appendix A.5 for detailed graphs). These performance levels represent drastic degradation from the perfor-

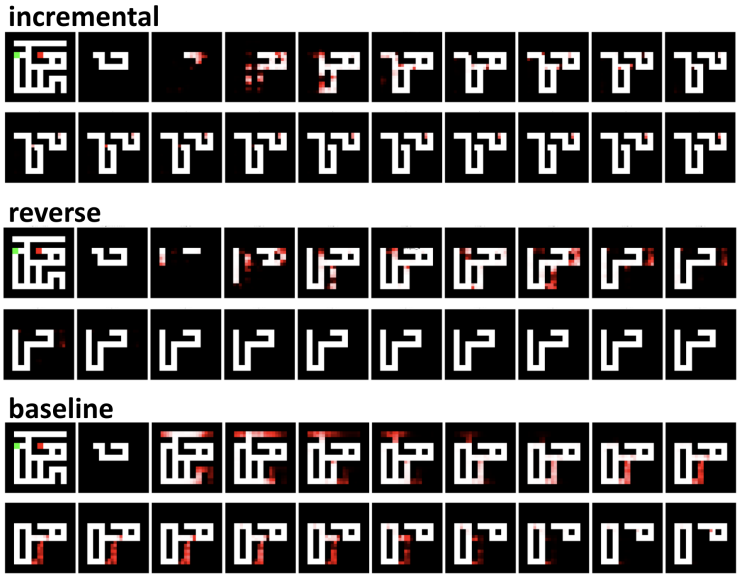


Figure 6: Representative failure cases of network generalisation on 9×9 test mazes with $p = 0.1$

mance on standard tree-structured mazes. This failure occurred when the maze size remained identical to the training conditions, suggesting that the learned strategies were as dependent on structural assumptions as the baseline model.

Representative failure cases can be seen in Figure 6. Networks settle with confidence on predictions that are either completely incorrect (the path is not connected from start to goal) or suboptimal (the path connects start to goal but does not correspond to the shortest path).

3.4 Limitations and Methodological Considerations

Two methodological limitations merit discussion. **First**, while the intermediate reasoning traces provide spatial memory through pixel-based representations, the recurrent CNN architecture lacks the structured memory mechanisms that classical maze-solving algorithms rely upon. For example, algorithms like A* require priority queues to maintain frontier nodes sorted by cost, depth-first search requires stacks to track backtracking paths, and breadth-first search requires queues for tracking explored nodes. These discrete data structures containing node coordinates, path costs, visited sets, and exploration priorities cannot be encoded within the pixel-based image representations that our CNN architecture processes. The current approach constrains all reasoning to spatial patterns in image pixels, forcing the models to learn spatial approximations of search algorithms and potentially explaining the observed brittleness under domain shifts. **Second**, due to computational budget constraints, the experiments did not include comprehensive ablation studies for two hyperparameters. We did not systematically explore the balance parameter α, which controls weighting between final accuracy and intermediate supervision adherence. Similarly, we only limitedly investigated the $step$ parameter (fixed at $step = 3$) governing temporal granularity of supervision traces. The relationship between trace density and learning outcomes remains an open question.

4 Discussion

4.1 Implications for Implicit Reasoning in Looped Architectures

Our results present a consistent pattern: networks that receive explicit procedural guidance learn to imitate the prescribed strategies faithfully, yet this imitation does not translate into robust algorithmic reasoning. The end-to-end baseline, which receives no intermediate supervision and develops its own implicit computational strategy, outperforms all supervised approaches on every generalisation measure. This suggests that explicit intermediate supervision constrains the network's representa-

tional freedom, forcing it to allocate capacity to reproducing a prescribed external procedure rather than discovering internal representations optimised for the task. The supervised networks learn *what the algorithm looks like* rather than *what the algorithm achieves*.

This finding points to the tension between explicit and implicit reasoning in neural networks. The recurrent CNN architecture implements a form of looped computation in which the network iteratively refines its internal state, and the end-to-end baseline exploits this loop to develop its own latent reasoning strategy. This latent strategy, while not easily interpretable as a classical algorithm, proves more robust than the explicit strategies we attempted to teach. In other words, implicit "thinking" seems to outperform explicit reasoning traces in this domain.

These findings may carry implications for chain-of-thought (CoT) training in large language models. If explicit step-by-step supervision undermines generalisation even in a fully controlled setting with algorithmically grounded traces, we should be cautious in assuming that CoT training induces robust reasoning in more complex domains. The consistent failure across four algorithmically distinct strategies suggests the limitation may be inherent to explicit supervision rather than specific to any particular reasoning trace design.

From a safety perspective, the confident convergence on incorrect solutions is troubling. The supervised models do not exhibit uncertainty when generalising poorly. Instead, they produce outputs with the same confidence as during training, despite solving problems incorrectly. Making the reasoning process explicit does not guarantee correctness, and may create a false sense of reliability when systems confidently follow learned patterns that fail outside their training distribution.

4.2 Related work

Brittleness of Reasoning Models Other recent work has revealed brittleness in networks that appear to exhibit logical extrapolation. Knutson et al. (2025) investigated maze-solving networks, testing both recurrent networks (Bansal et al., 2022) and implicit neural networks (INN), a class of architectures whose outputs are fixed points of neural network operators (Anil et al., 2022). Despite architectural diversity, both network types trained on simple maze navigation tasks failed when evaluated across variations in maze topology and goal configurations. Performance collapsed for both RNNs and INNs despite success on size extrapolation, suggesting that successful generalisation was more limited than previously claimed. These findings established that neither dynamic computation through recurrence nor fixed point computation through implicit networks automatically translated to robust logical extrapolation. However, Knutson et al. (2025) examined networks trained with end-to-end supervision rather than explicit intermediate guidance, leaving open the question of whether process supervision might address these limitations across different architectures. We address this gap by demonstrating that intermediate supervision not only fails to mitigate these brittleness issues but may actually introduce additional constraints that limit network flexibility.

Search-Based Interpretations of Neural Reasoning Several lines of work have proposed parallel reasoning mechanisms to overcome the limitations of greedy chain-of-thought decoding. In language models, approaches such as Tree-of-Thoughts explicitly implement parallel reasoning as search over multiple textual trajectories (Yao et al., 2023). By contrast, some latent-reasoning approaches argue that parallelism can emerge implicitly in continuous hidden states without explicit branching, and have interpreted this behaviour as breadth-first–like exploration in latent space (Hao et al., 2025). Our findings suggest an important distinction between these views: when parallel refinement is made explicit and enforced in a domain with ground-truth structure, it does not recover the benefits observed in unconstrained latent reasoning, indicating that latent parallelism reflects different continuous dynamics than those captured by explicit search in discrete spaces.

Limitations of Process Supervision Stechly et al. (2025) provided a direct challenge to semantic interpretations of reasoning traces by testing process supervision in maze pathfinding tasks using Transformer architectures. Their investigation revealed that meaningless reasoning traces outperformed semantically correct ones, achieving 26% accuracy versus 2.5% for correct traces on out-of-distribution pathfinding problems, challenging the assumption that intermediate supervision teaches networks to follow meaningful reasoning procedures. The specific findings on maze reasoning align with broader evidence from other domains, suggesting that intermediate supervision can hinder rather than help learning. Frydenlund (2025) demonstrated that decoder-only language models fail

to achieve better than chance performance when provided with intermediate waypoint supervision in graph search tasks, showing that excessive supervision prevents learning simple tasks. This work provided mechanistic insights into how intermediate supervision can interfere with learning, complementing the semantic interpretation challenges observed in maze domains.

4.3 Future Work

We see two important directions that deserve further investigation. **First**, training a single model on a diverse curriculum of search strategies could test whether broader supervisory signals encourage more generalisable reasoning. This would reveal whether a more varied supervisory signal could foster the development of more adaptable reasoning capabilities. **Second**, following Stechly et al. (2025), training with noisy (e.g. depicting suboptimal search paths) or randomised traces would directly test whether semantic content matters, or whether intermediate representations function purely as a computational scaffold. This would address the question posed by Knutson et al. (2025) regarding the conditions under which reliable algorithm learning is possible, helping to indicate when intermediate supervision truly imparts algorithmic understanding versus merely providing a brittle crutch.

5 Conclusion

This work investigated whether intermediate supervision enables neural networks to acquire robust algorithmic reasoning for maze-solving. Networks trained with step-by-step guidance successfully replicated prescribed search strategies on in-distribution data, but all intermediate supervision approaches underperformed the end-to-end baseline across every generalisation dimension tested. Under topological domain shifts, accuracy collapsed to less than 0.1% for all approaches, revealing a failure to learn general algorithmic principles. These results suggest that explicit procedural guidance constrains rather than enables robust reasoning, and that intermediate supervision may encourage networks to learn heuristics that are tightly coupled to the training data's structural properties rather than generalisable algorithmic logic.

REFERENCES

Cem Anil, Ashwini Pokle, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, J Zico Kolter, and Roger B Grosse. Path independent equilibrium models can better exploit test-time computation. *Advances in Neural Information Processing Systems*, 35:7796–7809, 2022.

Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Logical extrapolation without overthinking, 2022. URL https://arxiv.org/abs/2202.05826.

Kit Mills Bransby, Arian Beqiri, Woo-Jin Cho Kim, Jorge Oliveira, Agisilaos Chartsias, and Alberto Gomez. Backmix: Mitigating shortcut learning in echocardiography with minimal supervision, 2024. URL https://arxiv.org/abs/2406.19148.

Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

Arvid Frydenlund. Language models, graph searching, and supervision adulteration: When more supervision is less and how to make more more, 2025. URL https://arxiv.org/abs/2503.10542.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2025. URL https://arxiv.org/abs/2412.06769.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Michael Igorevich Ivanitskiy, Rusheb Shah, Alex F. Spies, Tilman Räuker, Dan Valentine, Can Rager, Lucia Quirke, Chris Mathwin, Guillaume Corlouer, Cecilia Diniz Behn, and Samy Wu Fung. A configurable library for generating and manipulating maze datasets, 2023. URL http://arxiv.org/abs/2309.10498.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Brandon Knutson, Amandin Chyba Rabeendran, Michael Ivanitskiy, Jordan Pettyjohn, Cecilia Diniz-Behn, Samy Wu Fung, and Daniel McKenzie. On logical extrapolation for mazes with recurrent and implicit networks, 2025. URL https://arxiv.org/abs/2410.03020.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks, 2021. URL https://arxiv.org/abs/2106.04537.

Kaya Stechly, Karthik Valmeekam, Atharva Gundawar, Vardhan Palod, and Subbarao Kambhampati. Beyond semantics: The unreasonable effectiveness of reasonless intermediate tokens, 2025. URL https://arxiv.org/abs/2505.13775.

Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. The clrs algorithmic reasoning benchmark. In *International Conference on Machine Learning*, pp. 22084–22102. PMLR, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

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. URL https://arxiv.org/abs/2305.10601.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

A IMPLEMENTATION DETAILS

A.1 NETWORK ARCHITECTURE

We use recurrent CNNs that can perform iterative computation through iterative forward passes. A network is defined as f_θ that operates on RGB maze representations $M \in \mathbb{R}^{H\times W\times 3}$, where red (1,0,0) and green (0,1,0) encode the start and goal positions $s, g \in \{1,\dots,H\}\times\{1,\dots,W\}$. These positions are marked directly in the input image. The networks produce binary path predictions $\mathbf{y} \in \{0,1\}^{H\times W}$ indicating the optimal solution path.

The architecture in Bansal et al. (2022) builds upon ResNet foundations (He et al., 2016), but uses a fixed-size residual block (called *recurrent module*) that gets repeated for T iterations. Weights are shared between these residual blocks. The input encoder $P_{\theta_P} : \mathbb{R}^{H\times W\times 3} \to \mathbb{R}^{H\times W\times D}$ is composed of one single convolutional layer with kernel size 3 and ReLU activation, and it projects the three-channel RGB maze representation into a higher-dimensional feature space of width D. Each recurrent block $A_{\theta_A} : \mathbb{R}^{H\times W\times(D+3)} \to \mathbb{R}^{H\times W\times D}$ then contains four convolutional layers with ReLU activation. This recurrent block is the main computational component, which processes features iteratively while maintaining access to the original RGB input through a recall mechanism (a residual connection from the input to each layer) that prevents information degradation. The output decoder $H_{\theta_H} : \mathbb{R}^{H\times W\times D} \to \mathbb{R}^{H\times W\times 2}$ transforms the refined features into per-pixel logits.

Figure 1b shows the complete network computation, which unfolds through T iterations. At each step t, the system produces both an intermediate output $\mathbf{O}_t \in \mathbb{R}^{H\times W\times 2}$ and an internal state $\mathbf{h}_t \in \mathbb{R}^{H\times W\times D}$ according to:

$$\begin{aligned}\mathbf{h}_t &= A_{\theta_A}(\mathbf{h}_{t-1}, M)\\ \mathbf{O}_t &= H_{\theta_H}(\mathbf{h}_t)\end{aligned}$$

where $\mathbf{h}_0 = P_{\theta_P}(M)$ provides the initial feature representation. This enables dynamic computational depth adjustment at test time, allowing networks trained with fixed iteration budgets to allocate additional computational resources for harder problems that require more computation time to reach a stable solution.

The concatenation of the current internal state with the original maze image at each iteration performed by the recurrent block prevents the information degradation observed in Schwarzschild et al. (2021) by ensuring that the input remains accessible throughout the reasoning process.

The three channels of the input encode different information: black pixels $(0,0,0)$ represent walls, white pixels $(1,1,1)$ represent traversable paths, red pixels $(1,0,0)$ mark the start position, and green pixels $(0,1,0)$ mark the goal position.

The two output channels represent raw logits for per-pixel classification as either path or background. During training, cross-entropy loss is computed between the raw logits $\mathbf{O}_t$ and the binary target path $\mathbf{y}_{\text{target}} \in \{0,1\}^{H\times W}$. At test time the final per-pixel prediction, $\mathbf{y} \in \{0,1\}^{H\times W}$, is obtained by applying argmax along the channel dimension, selecting the class with the highest logit.

Network Size and Hyperparameters The input encoder P_{θ_P} consists of a single 3×3 convolutional layer with stride 1 and padding 1, transforming the RGB maze input from 3 channels to the network width D. This layer is followed by ReLU activation and projects the maze representation into the feature space. The recurrent core A_{θ_A} implements a four-layer residual block following standard ResNet design (He et al., 2016). Each residual block contains two 3×3 convolutional layers with batch normalization omitted to maintain compatibility with recurrent computation. The block includes skip connections and ReLU activations. We extend the standard DeepThinking framework by introducing an additional input concatenation mechanism. The original recall mechanism concatenates the current hidden state with the input maze $[\mathbf{h}_{t-1}, M]$, and for these experiments the previous iteration's output prediction $\mathbf{O}_{t-1}$ is also concatenated with the input at each step. This creates the full recurrent input $[\mathbf{h}_{t-1}, M, \mathbf{O}_{t-1}]$, allowing the network to explicitly condition its reasoning on previous predictions. Additional convolutional layers map these concatenated inputs to maintain the feature dimensionality D. The output decoder H_{θ_H} comprises three sequential convolutional layers: the first two layers use 3×3 kernels with ReLU activation, progressively reducing the channel dimension, while the final layer produces 2-channel logits for binary path classification.

Table 2: Network specifications for maze experiments

Component	Specification
Network Width (D)	128
Input Encoder	3×3 Conv, $3 \rightarrow 128$ channels, ReLU
Recurrent Block	4-layer ResNet block, 128 channels
Output Decoder	3×3 Conv ($128 \rightarrow 32$), 3×3 Conv ($32 \rightarrow 8$), 3×3 Conv ($8 \rightarrow 2$)
Recall Concatenation	3×3 Conv for $[\mathbf{h}_{t-1}, M]$ concatenation
Output Concatenation	3×3 Conv for $[\mathbf{h}_{t-1}, M, \mathbf{O}_{t-1}]$ concatenation

A.2 Intermediate Supervision Generator

The intermediate supervision framework is implemented through a custom `MazeSolver` class that generates algorithm-specific supervision traces. The maze representation uses a patch-based encoding where each patch corresponds to a 2×2 pixel region in the input image.

The solver operates on RGB mazes with the following encoding: walls $(0, 0, 0)$, free paths $(1, 1, 1)$, start positions $(1, 0, 0)$, and goal positions $(0, 1, 0)$. Input preprocessing includes cropping the outermost pixel ring to remove boundary artifacts and converting the RGB representation to a binary free-space mask on the patch grid. Each supervision mask is generated at the patch level and then mapped back to full pixel resolution through 2×2 upsampling, ensuring correspondence with the network's spatial predictions.

The supervision generator includes a configurable step parameter $step$ that controls temporal granularity, allowing adjustment of trace density. For all experiments reported here, the step parameter is fixed at $step = 3$, which provides a balance between computational efficiency and sufficient temporal resolution for the model to learn meaningful intermediate reasoning steps. An example of the difference in reasoning traces produced by varying the $step$ parameter can be seen in Figure 7. In addition to what is seen in Figure 7, final frames are duplicated multiple times to ensure stable convergence signals during training.

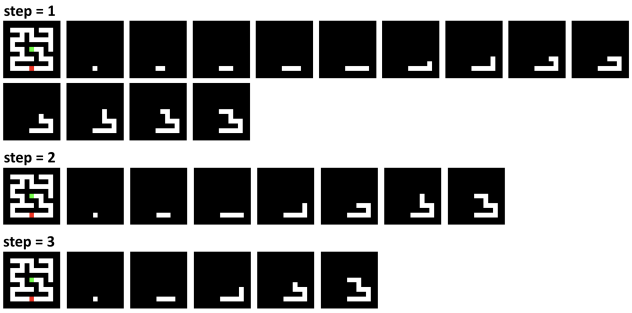


Figure 7: Supervision reasoning traces generated for incremental strategy with $step \in (1, 2, 3)$.

Each search strategy implements a distinct approach developed to be visually interpretable, as described in Section 2.

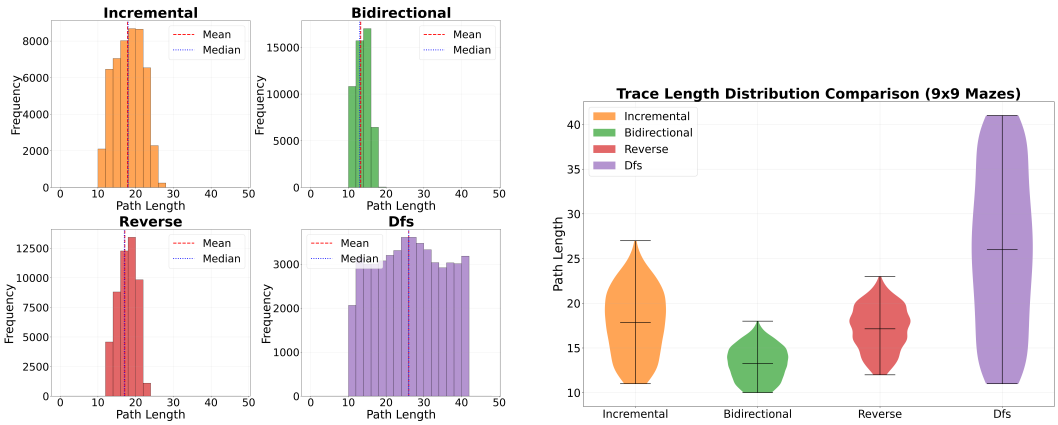
Statistical analysis of supervision trace lengths across the training dataset reveals significant variation both between and within strategies. As shown in Table 3, the mean trace lengths for 9×9 mazes range from 13.26 steps for bidirectional search to 25.99 steps for dfs. The bidirectional approach produces the most consistent traces with the lowest standard deviation (1.89), while dfs exhibits

the highest variability (standard deviation of 8.88) due to its exploration of multiple dead-end paths before backtracking.

Table 3: Supervision trace statistics across algorithmic strategies on 9×9 mazes.

	Incremental	Bidirectional	Reverse	Dfs
Mean	17.85	13.26	17.14	25.99
Median	18.00	13.00	17.00	26.00
Std	3.75	1.89	2.50	8.88
Min	11.00	10.00	12.00	11.00
Max	27.00	18.00	23.00	41.00
Q1	15.00	12.00	15.00	19.00
Q3	21.00	15.00	19.00	33.00

The distribution analysis in Figures 8 further illustrates these differences: incremental and bidirectional strategies produce shorter, more focused traces suitable for direct path learning, while reverse exploration and dfs generate longer sequences that emphasise exploration and pruning behaviours. Notably, high variation can be observed even within individual strategies, as evidenced by the substantial range between minimum and maximum values (e.g., dfs traces varying from 11 to 41 steps).



(a) Histograms of supervision trace lengths by strategy.

(b) Violin plot of supervision trace lengths by strategy.

Figure 8: Distribution of supervision trace lengths for different strategy types.

This variation in trace length is accommodated through the temporal normalisation mechanism in the supervision loss (Equation 2.2), which ensures that training samples with different supervision trace lengths can be trained effectively within the same batch.

The choice of $T_{\max} = 1000$ for the maximum iteration limit in the stable testing protocol was determined through analysis of the computational requirements across different search strategies and maze sizes. An analysis was conducted on the maximum trace lengths produced by each algorithmic strategy across the full range of test maze sizes to ensure adequate computational budget for convergence.

The scaling behaviour of trace lengths with maze size demonstrates substantial variation across strategies, as shown in Figure 9. While bidirectional and incremental strategies exhibit relatively modest growth patterns, dfs shows dramatic increases in computational requirements, particularly for larger maze instances. The reverse exploration strategy demonstrates intermediate scaling behaviour.

The maximum observed trace lengths across all strategies and sizes informed our selection of $T_{\max} = 1000$, providing sufficient computational budget to accommodate the longest reasoning traces. The considerable standard deviation observed across strategies, particularly for larger maze sizes, underscores the importance of allowing variable computational depth during evaluation. The

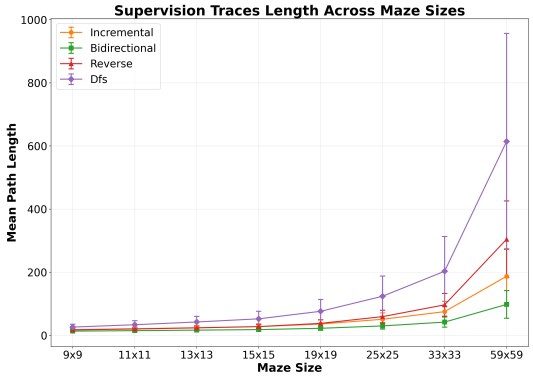


Figure 9: Mean path length of supervision traces across varying maze sizes for different strategies.

Table 4: Mean path lengths by size and algorithm across test datasets.

Size	Bidirectional	Dfs	Incremental	Reverse
9×9	13.24	26.05	17.82	17.15
11×11	14.71	33.44	20.75	19.96
13×13	16.33	42.21	23.99	23.51
15×15	18.06	52.14	27.46	27.57
19×19	22.18	76.05	35.69	37.79
25×25	29.74	123.83	50.81	59.18
33×33	41.82	202.97	74.97	96.52
59×59	97.79	614.24	186.92	304.18

stable testing protocol addresses this by enabling early termination when convergence criteria are met, while the maximum iteration limit ensures computational efficiency in cases where networks fail to reach stable solutions.

A.3 Training Hyperparameters

Training employs the Adam optimizer (Kingma & Ba, 2014) with most hyperparameters taken from Bransby et al. (2024)

The supervision balance parameter α is determined through experimentation across different values, with selection based on validation performance to ensure balance between final accuracy and intermediate supervision.

A.4 Computational Resources

We conducted all experiments using dual NVIDIA GeForce RTX 2080 Ti GPUs (11GB VRAM each). Training a network on 9x9 mazes required between 3 and 8 hours, depending on the strategy. The time required to test a network also depended on strategy but mainly on the size of the test mazes (Figure 10). Memory requirements scale with the maximum iteration limit $T_{\max} = 1000$. GPU utilisation typically ranged between 80-95% during training.

A.5 Extended Results

Table 7 shows the full set of evaluation results across all models and maze sizes.

Figure 11 shows the collapse of models accuracies when tested on loopy mazes.

Table 5: Standard deviations by size and algorithm across test datasets.

Size	Bidirectional	Dfs	Incremental	Reverse
9×9	1.89	8.92	3.76	2.52
11×11	2.64	13.04	5.25	3.74
13×13	3.44	17.97	6.86	5.24
15×15	4.39	23.85	8.77	7.08
19×19	6.51	37.73	13.01	11.48
25×25	10.12	63.88	20.24	20.18
33×33	16.23	110.20	32.45	36.09
59×59	43.87	342.46	87.75	121.56

Table 6: Training configuration

Parameter	Value
Optimizer	Adam
Learning Rate	0.001
Learning Schedule	StepLR
Weight Decay	0.0001
Batch Size	32
Validation Split	20%

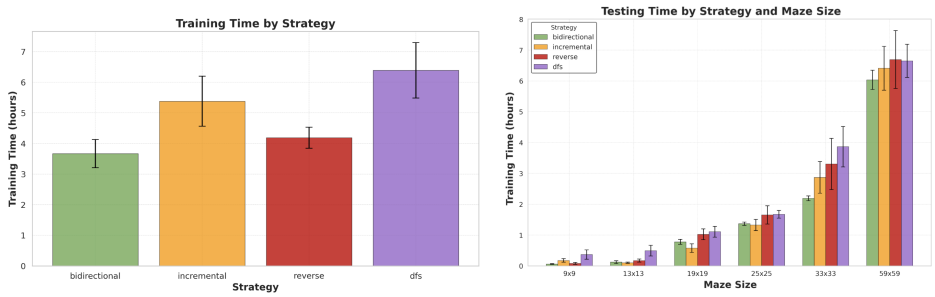


Figure 10: Training and testing runtimes across strategies.

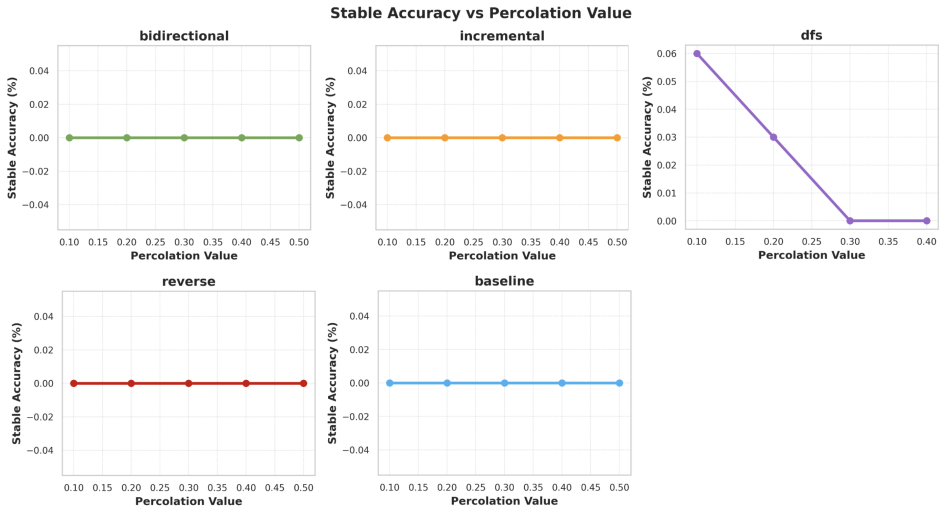


Figure 11: Mean stable test accuracy achieved on 9×9 mazes after $T_{max} = 200$ iterations by networks trained with different search strategies on mazes with increasing values of percolation p.

Table 7: Summary statistics over ten training and testing runs (mean, standard deviation, and standard error) for each model grouped by maze size.

Size	Model	Mean	Std	SE
9x9	baseline	95.26	10.58	3.347
	bidirectional	99.95	0.1141	0.04033
	dfs	79.61	35.86	12.68
	incremental	99.86	0.3527	0.1247
	reverse	99.98	0.02806	0.008460
13x13	baseline	92.44	16.12	5.097
	bidirectional	98.63	2.489	0.8798
	dfs	73.07	43.46	15.37
	incremental	98.61	2.295	0.8114
	reverse	96.19	7.470	2.252
19x19	baseline	88.05	20.98	6.634
	bidirectional	82.83	12.29	4.346
	dfs	60.84	39.92	14.11
	incremental	88.90	13.83	4.891
	reverse	64.91	23.29	7.022
25x25	baseline	81.58	23.36	7.388
	bidirectional	54.29	16.76	5.925
	dfs	46.97	35.24	12.46
	incremental	72.72	25.32	8.951
	reverse	45.61	31.62	9.535
33x33	baseline	74.94	27.32	8.641
	bidirectional	28.71	15.27	5.397
	dfs	34.01	30.85	10.91
	incremental	53.23	32.87	11.62
	reverse	30.54	34.69	10.46
59x59	baseline	57.60	34.66	10.96
	bidirectional	7.061	5.625	1.989
	dfs	14.26	19.43	6.869
	incremental	21.19	21.09	7.458
	reverse	21.42	38.75	11.68