Accelerating A* Search with Learned Residual Heuristics in Path Planning

Introduction

- A* performance depends on the heuristic
- Classical heuristic (Manhattan) ignores obstacles
- Goal: integrate a learned heuristic into A*

A* Implementation & Integration

- Generic heuristic_func interface for flexibility
- LearnedHeuristic adapter:
    1. loads PyTorch model once
    2. converts (x, y) → 3-channel tensor
    3. reconstructs heuristic:

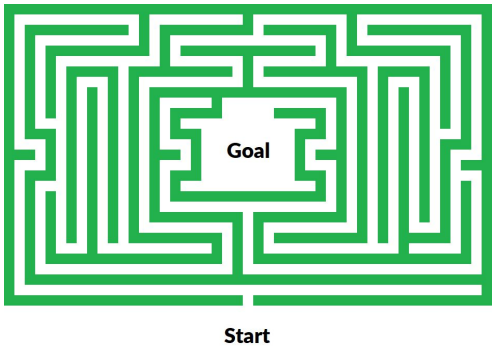
$$h(s) = d_{\text{Manhattan}}(s) + r(s)$$

Maze & Data Generation

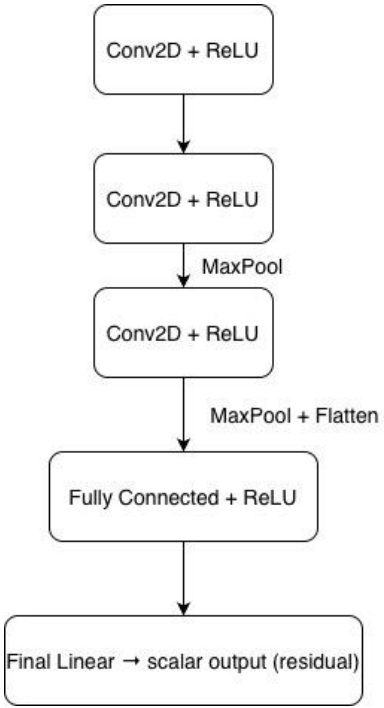
- 200 random **15×15** mazes
- Obstacle density: **10–40%**
- BFS provides exact shortest-path labels
- 3-channel state encoding:
    1. obstacles
    2. agent position
    3. goal position
- Residual target:

$$r(s) = d^*(s) - d_{\text{Manhattan}}(s)$$

- Tens of thousands of training samples



CNN structure

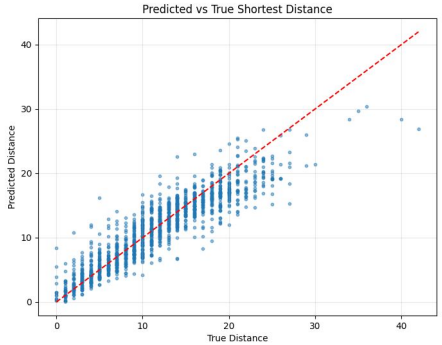


Parameters:

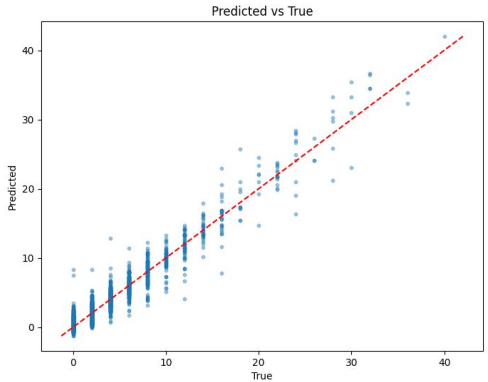
- Loss: **Mean Squared Error (MSE)**
- Optimizer: **Adam**
- Batch size: 64
- Epochs: 20
- Validation split: 20%

Performance:

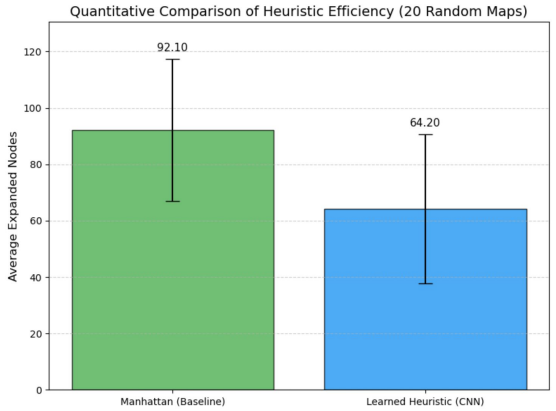
before:



after:



Evaluation & Results



Robust Testing Framework

- Automated random comparison loop
- Filters impassable maps: Guarantees 20 valid/reachable environments
- Ensures statistical reliability (no skew from invalid cases)

Quantitative Results

- Metric: Average Expanded Nodes (Lower is better)
- Baseline (Manhattan): ~ 92.1 nodes
- Learned (Residual): ~ 64.2 nodes
- Conclusion: ~30% reduction in search effort