Accelerating A* Search with Learned Residual Heuristics in Path Planning

Minzheng Song [*1] Wenyang Hu [*1] Xiyun Hu [*1] Xinyi Guo [*1]

Abstract

We investigate whether a lightweight learned heuristic can improve the efficiency of A* search in grid-based path planning. A convolutional neural network is trained on BFS-derived labels to predict residual corrections to the Manhattan heuristic, enabling obstacle-aware cost estimates with minimal computational overhead. Across unseen 15×15 mazes, the learned heuristic reduces node expansions by 30–40% while maintaining near-optimal path quality. These results show that compact residual models can effectively enhance classical A* search without sacrificing interpretability.

1. Introduction

Motion planning is a classic problem in robotics, game theory, and intelligent decision-making systems. Of several traditional search algorithms, A* is one of the most seminal, given its optimality with efficiency properties, which are attained via a heuristic function.

Classic heuristic methods like Manhattan distance or Euclidean distance are efficient, easy to calculate, and optimal. But their major drawback is that they lack any information about their environment. On complex maps containing obstacles, bottlenecks, long corridors, and dead ends, those heuristics could produce a drastically underestimated value for the shortest path between start and goal nodes.

But recent work in machine-learning offers a hint at something different: why not use learner algorithms to figure out heuristic functions based on their data? Learning heuristic functions could also enable us to represent complex patterns in grids in ways not previously possible with heuristics. Previous research has actually indicated that learning heuristic functions could be useful by helping to reduce costs through being more informed about cost to go.

[1]University of North Carolina at Chapel Hill, Chapel Hill, NC, USA. Correspondence to: Jorge Silva <jsilva@cs.unc.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

Within this project, we explore this idea by building an end-to-end A* planning system augmented with a learned heuristic. A CNN is trained to estimate residual corrections to the Manhattan distance using shortest-path labels generated by BFS. The resulting heuristic incorporates environment-specific information that classical heuristics overlook, allowing A* to make more informed decisions without increasing algorithmic complexity. We then compare the learned heuristic against the traditional Manhattan heuristic across a variety of randomly generated grid maps, examining differences in search effort, runtime, and qualitative expansion patterns. Our results show that even a compact neural model can capture meaningful structural properties of the environment and improve the efficiency of classical A* search. To summarize, our pipeline can be listed as following:

1. **Implement a standard A* search engine.** We construct the core A* algorithm with support for pluggable heuristic functions, priority-queue management, cost accumulation, and path reconstruction.

2. **Construct a grid/maze environment.** Maps are represented as binary $0/1$ grids, where 1 denotes obstacles.

3. **Train a convolutional neural network heuristic model.** Using BFS as the shortest-distance oracle, we produce supervised labels. The model predicts a residual correction to the Manhattan heuristic.

4. **Evaluate A* with classical and learned heuristics.** Metrics include expanded nodes, search performance, path quality, heuristic error, and visual comparisons of search behavior across grid maps.

2. Related Work

2.1. Classical Heuristic Search and A* Efficiency

A* Search: It was conceptualized in the late 1960s (Hart et al., 1968). Later works have ensured its optimality property formulation and established admissible heuristic values (Pearl, 1984; Dechter & Pearl, 1988). Using classical geometry or Manhattan, Euclidean, Chebyshev distances is admissible but rather simple and widely underestimated for any obstacles or corridors (Sturtevant, 2007; Rabin & Sturtevant, 2016).

Additionally, more advanced techniques such as pattern database heuristics (Culberson & Schaeffer, 1998) and techniques utilizing landmarks from automated planning (Hoffmann et al., 2004) have shown to effectively cut down on the number of searches using pre-computed information or structural knowledge. Those techniques may require significant memory or intricate engineering tasks for each domain and thus may also spark interest among researchers for data-driven heuristics capable of making generalizations for different settings.

2.2. Residual Learning in Predictive Models

Residual learning, as demonstrated by deep residual learning networks (He et al., 2016), is effective at making optimization easier because instead of learning everything from scratch, it tries to learn residual values to a reference estimate.

Additionally, neural network approaches have also been investigated for cost-to-go functions or for planning module integration into CNN architectures, e.g., Value Iteration Networks (Tamar et al., 2016). Other studies have also looked at learning heuristic functions based on state and distance pair examples (Arfaee et al., 2011; Barrett et al., 2013; Guez et al., 2019).

Residual heuristics represent a lightweight hybrid approach: instead of learning to predict full shortest path distances, one could learn to predict just a correction to some approximate classical heuristic like Manhattan distance. This should make these correction terms smoother and thus learnably approximateable, making it possible to learn heuristics informed by knowledge of obstacles while still having classical interpretability and stability.

3. Methodology

3.1. A* Implementation and Heuristic Integration

The core pathfinding engine in our system is built on a standard A* search implementation. Our primary contribution is the integration of a flexible heuristic interface that allows the classic Manhattan heuristic to be replaced seamlessly with a learned neural model.

3.1.1. A* Core and Adapter Design

The A* implementation uses a priority queue to maintain the open set, always selecting the node with the minimum

$$f(s) = g(s) + h(s).$$

To support both analytical and learned heuristics, the algorithm accepts a generic `heuristic_func` argument. This modular design enables the search routine to remain unchanged even when the source of heuristic estimates varies.

To bridge A* with the PyTorch model, we developed a dedicated `LearnedHeuristic` class. This component acts as an adapter between the abstract A* interface and the concrete requirements of the neural network. It handles two critical tasks:

- **Efficiency.** The class loads the model once and precomputes static elements such as the obstacle map and goal channel, minimizing overhead during the thousands of heuristic queries inside A*.
- **State Translation.** A* represents states as (x, y) coordinates. The learned model requires a $3 \times H \times W$ tensor. The adapter dynamically injects the agent's position into this representation.

3.1.2. Residual Heuristic Reconstruction

The neural network is trained to predict a residual correction term,

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

where $d^{(s)}$ is the BFS shortest-path distance. During search, the heuristic used by A* is reconstructed as

$$h(s) = d_{\text{Manhattan}}(s) + r_\theta(s).$$

This formulation combines the stability of the Manhattan heuristic with obstacle-aware corrections learned from data.

3.2. Data Generation and State Encoding

To train the neural heuristic, we generated a supervised dataset from 200 randomly sampled 15×15 mazes, with obstacle densities drawn uniformly from $[0.10, 0.40]$. This provides diverse map structures including open regions, corridor-like shapes, and maze-like bottlenecks.

3.2.1. Ground-Truth Distance Computation

For each maze, we computed the shortest-path distances to a fixed goal using Breadth-First Search (BFS). Because the grid is unweighted and 4-connected, BFS yields exact shortest-path distances efficiently. Unreachable states were excluded.

3.2.2. State Encoding

Each valid state is encoded as a 3-channel tensor:

- **Obstacle channel:** binary occupancy grid.
- **Agent channel:** one-hot encoding of the agent location.
- **Goal channel:** one-hot encoding of the target cell.

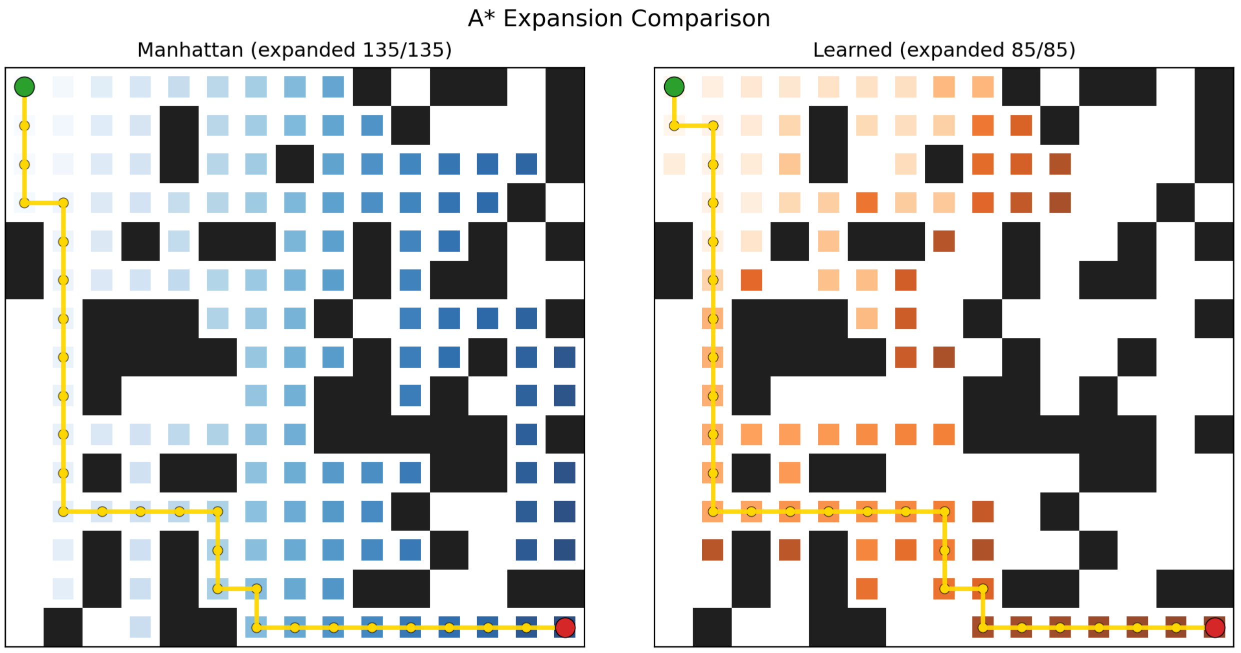

Figure 1. **Comparison of A* search behavior using Manhattan vs. learned residual heuristics.** The left panel shows node expansions under the Manhattan heuristic (135 expanded nodes), which produces a broad and diffuse frontier. The right panel shows the learned residual heuristic, resulting in a more focused search with only 85 nodes expanded. The learned model reduces unnecessary exploration while preserving the same final path, demonstrating more informative obstacle-aware guidance.

3.2.3. Residual Target Formulation

Instead of learning the full distance $d^{(s)}$, we use the residual target

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

which isolates obstacle-induced detours and simplifies learning.

3.2.4. Dataset Construction

All reachable free cells from each maze are encoded and paired with their residual targets. The aggregated dataset contains tens of thousands of samples, shuffled and split into 80% training and 20% validation sets, and stored in compressed `.npz` format.

3.3. Neural Network Training

3.3.1. Motivation

The Manhattan heuristic

$$d_{\text{Manhattan}}(s) = |x_s - x_g| + |y_s - y_g|$$

is admissible but often severely underestimates true distances in the presence of obstacles. Directly regressing $d^{(s)}$ proved challenging due to overestimation risks and large value ranges. Residual learning mitigates these issues.

3.3.2. Model Architecture

We employ a lightweight CNN:

- Conv2D $\rightarrow$ ReLU
- Conv2D $\rightarrow$ ReLU $\rightarrow$ MaxPool
- Conv2D $\rightarrow$ ReLU $\rightarrow$ MaxPool
- Flatten
- Fully connected $\rightarrow$ ReLU
- Linear output layer (scalar residual)

3.3.3. Training Procedure

Training uses Mean Squared Error (MSE) loss, Adam optimizer, batch size 64, and 20–30 epochs. A 20% validation split monitors generalization. We evaluate the model using MAE, error histograms, and predicted-vs-true scatter plots.

3.4. Using the Neural Heuristic in A*

For each state s expanded during search:

1. Encode the state into a $3 \times H \times W$ tensor.
2. Predict the residual: $r_\theta(s)$.

3. Reconstruct the heuristic:

$$h(s) = d_{\text{Manhattan}}(s) + r_\theta(s).$$

This learned heuristic replaces the classical Manhattan estimate during A*, providing obstacle-aware guidance while maintaining computational efficiency.

4. Experiment

4.1. Overview

The experiments are intended to determine whether the residual heuristic learned is able to improve A search efficiency over the classical Manhattan heuristic distances between unseen 15×15 mazes.

4.2. Dataset Characteristics

After preprocessing the 200 sampled mazes, the final dataset ended up having tens of thousands of reachable state observations. Residual values were between 0 and around 12, with most values being centered around 0. This showed that while localized areas have detours because of obstacles, most paths have near-zero residual values or are unobstructed pathways. The rates of reachability were dependent on the levels of obstacles to create a good distribution for training.

4.3. Training Dynamics

The CNN residual model converged rapidly, with training and validation loss stabilizing within 10–12 epochs. The performance on the validation set showed very low values for mean absolute error, good alignment between predicted values and actual values, and very small error distribution. These outcomes indicate that the model generalized well to the held-out validation set and learned to recognize structural patterns responsible for shortest-path deviations.

4.4. A* Performance Comparison

We evaluated A* using both Manhattan and learned residual heuristics on a separate test set of unseen mazes.

4.4.1. Quantitative Results

The learned heuristic consistently required fewer node expansions. Across all trials, the learned heuristic reduced node expansions by approximately 30–40% while maintaining comparable solution quality.

4.4.2. Path Quality

In all cases, the final paths obtained by both heuristics differed by at most 1 to 2 steps or were equivalent. This performance indicates that near-optimality was guaranteed after learning residuals without substantial residual-induced overestimation.

4.5. Visualization of Search Behavior

To better understand the behavioral differences, we visualized the expansion order of both heuristics.

4.5.1. Manhattan Heuristic

The Manhattan heuristic generated extensive and diffuse searching activities, exploring many nodes in directions which had walls or dead ends. This expansion area was large and did not contain any part of the optimal path.

4.5.2. Learned Heuristic

The learned heuristic allowed for more focused expansions around the region of interest centered on the actual solution region. Dead-end paths were avoided, curves occurred smoothly around obstacles, and efforts were focused around structurally interesting areas.

4.5.3. Comparison

Side-by-side heatmaps indicate that the Manhattan heuristic performs almost twice as many node expansions than those conducted by A*. The learned residual network removes large areas of irrelevant information while maintaining good paths, demonstrating more geometry-aware search behavior.

5. Conclusion

This work demonstrates that a simple learned residual heuristic can effectively speed up classical A* search on grid environments. A CNN-based approach to learning corrections to Manhattan distances makes the heuristic aware of obstacles while maintaining the robustness of the analytically derived heuristic.

In various test mazes, the learning heuristic resulted in around 30-40These observations demonstrate the utility of blending traditional searching algorithms with efficient data-driven modules to allow A to discover meaningful insights unobserved by rigid heuristics.

The approach is also limited by its rigid input size, use of synthetic training maps, and lack of admissibility guarantees. Possible future research directions include extending the approach to use variable-sized maps, considering a larger range of training distributions, or using techniques to maintain heuristic consistency.

In summary, learned residual heuristics have provided a viable approach to improve A* efficiency while maintaining interpretability and near optimality.

References

Arfaee, S. J., Zilles, S., and Holte, R. C. Learning heuristic functions for large state spaces. *Artificial Intelligence*, 175(16–17):2070–2098, 2011.

Barrett, T. W., Lee, S., and How, J. P. Learning heuristic functions for heuristic search. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*, 2013.

Culberson, J. C. and Schaeffer, J. Pattern databases. *Computational Intelligence*, 14(3):318–334, 1998.

Dechter, R. and Pearl, J. Generalized best-first search strategies and the optimality of a*. *Journal of the ACM*, 32(3): 505–536, 1988.

Guez, A., Weber, T., Antonoglou, I., and Silver, D. Investigating mcts and neural heuristics in small gridworlds. In *Deep Reinforcement Learning Workshop, NeurIPS*, 2019.

Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Hoffmann, J., Porteous, J., and Sebastia, L. Ordered landmarks in planning. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 174–181, 2004.

Pearl, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

Rabin, S. and Sturtevant, N. R. Suboptimal heuristics for suboptimal search. *Journal of Artificial Intelligence Research*, 57:635–685, 2016.

Sturtevant, N. R. Potential heuristics for path finding. In *Proceedings of the AAAI Workshop on Learning for Search*, 2007.

Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. Value iteration networks. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS)*, pp. 2154–2162, 2016.