IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, May 2025
Impact Factor: 7.67

Smart Path Finder via Neural A-Star

Manik Rajawat, Mohd Faisal Raeen, Keshav Gopal, Ms. Chitra
Information Technology
Raj Kumar Goel Institute of Technology, Ghaziabad, UP, India
rajawatmanik@gmail.com, edwardfaisal800@gmail.com,
keshavgopal2002@gmail.com, pgoelfcs@rkgit.edu.in

Abstract: *In today's fast-paced and ever-evolving technological landscape, intelligent navigation systems are increasingly vital across numerous domains—from robotics and autonomous vehicles to gaming and logistics. This paper presents the design and implementation of a Smart Path-Finding System leveraging a neural variant of the classic A* (A-Star) algorithm, aimed at enhancing real-time decision-making in dynamic environments.*

The proposed system integrates the classical heuristic efficiency of A with the predictive power of neural networks to address the limitations of traditional path-finding, particularly in environments that are large-scale, partially observable, or constantly changing. In this framework, a deep learning model is trained to approximate heuristic values by learning from previous navigation patterns across various map topologies. This enables the system to dynamically adjust path costs and anticipate better routes, rather than relying solely on static heuristic functions like Euclidean or Manhattan distances.*

The core architecture of the system includes a grid-based map environment, where obstacles and path costs can vary in real time. Data preprocessing involves encoding the environment into state representations, where each grid cell is translated into feature vectors representing obstacles, dynamic costs, and proximity to the goal.

To ensure robust performance, the system undergoes extensive simulation testing on both synthetic and real-world-like environments. Performance metrics include path optimality (compared to traditional A), time-to-solution, adaptability to environment changes, and computational efficiency.*

Keywords: Sentiment Analysis, speech-to-text technology, Machine Learning Models, Helpdesk Interactions and Business Impact

I. INTRODUCTION

In today's rapidly evolving technological landscape, intelligent navigation systems are essential for a wide range of applications including robotics, autonomous vehicles, logistics, and digital gaming. These environments are typically dynamic and require decision-making that is both fast and accurate. As the demand for adaptive, real-time path planning increases, traditional algorithms such as A* face limitations in handling complex, unpredictable scenarios efficiently.To address this challenge, the current project explores the development of a Smart Path-Finding System using a neural-enhanced variant of the A* algorithm, known as Neural A*.At its core, path-finding is a problem of navigating an agent from a source to a destination through the most optimal route, often represented on a grid or graph.

II. LITRATURE SERVEY

Hart, Nilsson, and Raphael (1968)

The foundational work on the A-Star algorithm was introduced by Peter Hart, Nils Nilsson, and Bertram Raphael in their seminal paper [1] "A Formal Basis for the Heuristic Determination of Minimum Cost Paths.". This algorithm combined the strengths of Dijkstra's algorithm and greedy best-first search to ensure both optimality and efficiency. A-Star relies on a cost function f(n)=g(n)+h(n)f(n) = g(n) + h(n)f(n)=g(n)+h(n), where g(n)g(n)g(n) is the path cost from the start node to node nnn, and h(n)h(n)h(n) is a heuristic estimate of the cost from nnn to the goal. This work laid the foundation for almost all heuristic-based pathfinding algorithms used today in robotics, games, and AI.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, May 2025
Impact Factor: 7.67

Silver et al. (2016)

In their groundbreaking work on AlphaGo, David Silver and his team at DeepMind highlighted [2] the power of neural networks to improve classical planning algorithms. While not directly about A-star, their research introduced the idea of using deep learning to guide search algorithms—particularly Monte Carlo Tree Search (MCTS). This inspired subsequent work that attempted to bring similar learning principles to other search algorithms like A-star. Their success with combining deep learning and traditional search techniques created the groundwork for developing Neural A-star variants.

Yang et al. (2021)

In the paper [4] "Neural A Star Search: Towards a Neural Version of Classical Planning," Yang and colleagues proposed a fully differentiable version of the A-Star algorithm, marking a significant evolution of the traditional approach. Their architecture includes a learned encoder and a differentiable A-Star planner that enables backpropagation through the planning steps. They demonstrated that their model could significantly reduce the number of nodes explored compared to traditional A-Star, while still maintaining near-optimal path lengths. This work is one of the most direct and thorough comparisons between learned and classical planning strategies.

Bhardwaj et al. (2017)

The paper [5] "Learning Heuristics for Search-based Planning" by Bhardwaj and colleagues focuses on using imitation learning to train neural networks that predict effective heuristics for A*-style planners. Their work showed that machine-learned heuristics can outperform traditional handcrafted heuristics, especially in high-dimensional or dynamic environments. This marked a crucial step toward hybrid planning systems like Neural A*, which use learning not to replace but to enhance classical planning.

Matplotlib Visualization Library (2023)

Matplotlib is a widely used Python [8] library for 2D plotting and data visualization, originally developed by John D. Hunter in 2003. It provides tools for creating static, animated, and interactive graphs, making it essential for projects like A* vs Neural A* algorithm comparisons. With support for libraries like NumPy and Pandas, it efficiently handles data and generates performance charts and path visualizations. Its customization options—such as colors, labels, and legends—make it effective for clear and accurate data presentation. Due to its simplicity and flexibility, Matplotlib remains a key tool for visualizing complex results in research and development.

Streamlit Framework (2023)

Streamlit is an open-source Python framework [7] for creating interactive web apps with minimal code. It lets users build data science dashboards without frontend skills. In this project, it enables maze configuration, real-time visualization, and CSV downloads. Its simplicity and integration with libraries like Matplotlib make it ideal for showcasing algorithm performance.

Chen et al. (2020)

In "Learning Heuristic Functions [3] for Search-Based Planning," Chen et al. developed a neural network thatlearnsheuristics to guide A* search more efficiently. This direct enhancement of A* using deep learning proved that machine-learned heuristics can dramatically reduce the search space while still preserving near-optimal solutions. Their work was influential in demonstrating practical benefits in real-time robotics and navigation.

Zico Kolter & Pieter Abbeel (2010)

In "Learning Heuristics for A-Star Search [6] using Imitation Learning," Kolter and Abbeel explored using imitation learning to derive heuristics for A* that outperform hand-crafted ones. This early work showed how learning-based methods could improve classical search efficiency and laid the foundation for integrating neural components into heuristic design.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, May 2025
Impact Factor: 7.67

III. METHODOLOGY

A-Star Algorithm

The A* (A-Star) algorithm is a widely used pathfinding and graph traversal technique known for its balance of performance and accuracy. It is commonly applied in navigation systems, robotics, and AI, especially in environments represented as grids or graphs. A* is designed to find the most cost-effective path from a start node to a goal node while minimizing the number of nodes explored.

A* maintains a collection of nodes to be explored using a priority queue, typically implemented as a min-heap. This queue, often referred to as the open list, ensures that nodes with the lowest estimated total cost are explored first. Each node's priority is determined by the cost function:

f(n)=g(n)+h(n)f(n) = g(n) + h(n)f(n)=g(n)+h(n)

where:

g(n)g(n)g(n) is the actual cost from the start node to the current node nnn,

h(n)h(n)h(n) is the heuristic estimate of the cost from nnn to the goal node.

For grid-based maps, the heuristic function often used is the Euclidean distance, which calculates the straight-line distance between the current node and the target node. This heuristic is admissible (never overestimates the cost), ensuring that the algorithm will always find the shortest path if one exists.

A* explores nodes in four cardinal directions—up, down, left, and right—from each current position. This movement model is ideal for grid-based maps and simplifies the implementation. During expansion, the algorithm checks each neighbouring cell and calculates its cost. If the cell is an obstacle or already visited with a lower cost, it is ignored.

One of A*'s key strengths is its ability to avoid obstacles intelligently. When implemented correctly, it ensures that only passable paths are considered and that impassable or blocked cells are skipped. As it proceeds, A* continues selecting the most promising node from the priority queue, expanding it, and updating neighbouring nodes' scores until the goal is reached.

In conclusion, A* combines the efficiency of greedy algorithms (through the heuristic) and the guaranteed optimality of Dijkstra's algorithm (through full cost tracking), making it both fast and reliable for shortest-path search in static environments.

Neural A* Algorithm:

The Neural A* algorithm is an extension of the classical A* search method that incorporates deep learning—particularly convolutional neural networks (CNNs)—to learn and adapt the heuristic function. While traditional A* relies on a manually crafted heuristic such as Euclidean or Manhattan distance, Neural A* learns this heuristic from data, allowing it to make more informed decisions in complex or previously unseen environments.

In Neural A*, the goal remains the same: to find the shortest or most efficient path from a starting point to a destination while avoiding obstacles. However, instead of computing the heuristic h(n)h(n)h(n) using a fixed mathematical formula, it uses a learned function, typically parameterized by a CNN trained on sample maps and optimal paths. The network takes as input a representation of the environment—such as an occupancy grid map, with start and goal positions—and outputs a heuristic value for each node.

This learned heuristic is then plugged into the same cost function used in A*:

f(n)=g(n)+h^(n)f(n) = g(n) + \hat{h}(n)f(n)=g(n)+h^(n)

Where:

g(n)g(n)g(n) is the true cost from the start to the current node,

h^(n)\hat{h}(n)h^(n) is the neural heuristic estimate, produced by the CNN.

By learning from data, Neural A* can capture more contextual information about the environment than traditional heuristics. For example, it can implicitly learn which areas are likely to be dead ends or more difficult to traverse, which would otherwise require additional hard-coded logic or domain knowledge. The use of convolutional layers allows the model to identify spatial patterns in the map—such as corridors, bottlenecks, or open spaces—improving heuristic accuracy.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, May 2025
Impact Factor: 7.67

A major advantage of Neural A* is its ability to generalize to unseen environments. Once trained, the model can be deployed on new, unseen maps and still produce effective heuristics, enabling faster and more efficient pathfinding. This is particularly useful in domains like robotic navigation or autonomous driving, where encountering novel scenarios is common.

Moreover, Neural A* often explores fewer nodes than traditional A*, thanks to its more informed heuristic. This leads to faster planning without sacrificing path quality. However, the performance of Neural A* is tightly coupled with the quality of the training data and the representational power of the neural network architecture.

In conclusion, Neural A* maintains the core mechanics of A* but replaces the static heuristic with a learnable, adaptive function. This blend of classical search structure with modern learning techniques results in a hybrid system that can scale to more complex, dynamic environments and serve as a bridge between traditional algorithms and end-to-end deep learning systems.

Performance Metrics: -

To evaluate the effectiveness and efficiency of pathfinding algorithms, it is essential to consider objective performance metrics. In this study, three key metrics are used to compare the classical A* algorithm with the Neural A* variant: **path length, computation time,** and **number of nodes expanded**. These metrics provide a comprehensive understanding of the trade-offs between optimality and computational efficiency.

Path Length: -

Path length refers to the total distance or number of steps taken from the start node to the goal node along the computed path. In grid-based maps, this is typically measured as the sum of movement costs between adjacent nodes. An optimal path is the one with the minimum possible length. This metric reflects the **quality of the solution** produced by the algorithm. While classical A* is guaranteed to find the shortest path if the heuristic is admissible, Neural A* aims to achieve similar optimality even with a learned heuristic. Any deviation in path length helps assess how close Neural A* comes to optimal performance.

Computation Time: -

Computation time is the total time taken by the algorithm to compute a valid path from start to goal. It includes the time spent on heuristic evaluation, node expansion, and decision-making. This metric is particularly important in **real-time applications** such as robotics or autonomous navigation, where timely responses are critical. Classical A* can become slow in large or complex maps due to exhaustive node expansion. Neural A*, by using a learned heuristic, may guide the search more intelligently, potentially **reducing runtime** while maintaining acceptable path quality.

Number of Nodes Expanded: -

This metric counts the total number of nodes explored (or expanded) during the search process. Each node represents a possible position in the environment that the algorithm considers on the way to finding the goal. A lower number of node expansions generally indicates a more efficient search strategy, as fewer resources are used to reach the destination. Traditional A* tends to expand many nodes, especially with poor heuristics. In contrast, a well-trained Neural A* model can significantly reduce node.

IV. TECHNOLOGY USED

This project utilizes a set of powerful Python-based tools and libraries to develop, evaluate, and present the comparison between the A* and Neural A* algorithms. Each technology plays a specific role in ensuring robust algorithm performance, clear visualization, and interactive usability.

1. Python Programming Language

Python serves as the foundation for the entire project due to its simplicity and extensive ecosystem. It supports everything from implementing algorithms and neural networks to building user interfaces and data visualization tools.

2. **NumPy and Pandas**

NumPy is used for efficient numerical operations and matrix manipulations, which are essential for representing the maze grid and managing search space data.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, May 2025
Impact Factor: 7.67

Pandas handles performance metrics, data collection, and exporting results in CSV format for further analysis and record-keeping.q

3. Matplotlib

Matplotlib is used to create various visualizations, including:Comparative path diagrams for A-Star and Neural A-Star Bar charts representing performance metrics (e.g., computation time, nodes expanded)Heatmaps and node expansion plots (if applicable)These visualizations help in understanding algorithm behavior and efficiency.

4.scikit-learn

scikit-learn is used for preprocessing and managing datasets for training the neural network. It aids in:Splitting data into training and validation setsScaling or normalizing features, if necessary,applying utility functions such as accuracy scoring and data shufflingIts simplicity and compatibility with other ML libraries make it ideal for preparing the data used in Neural A-Star's learning phase.

5. **TensorFlow / PyTorch**

This library is used to define and train the Convolutional Neural Network (CNN) that serves as the heuristic predictor in Neural A*. It provides tools for:Building and compiling neural architecturesTraining the model using supervised learningPerforming predictions during pathfinding in unseen mazesGPU support and model evaluation tools enhance performance and flexibility.

6. Streamlit

Streamlit provides the graphical interface for user interaction. It enables:Dynamic maze size and obstacle configurationAlgorithm selection and executionReal-time visualization of search and pathCSV file download of resultswith minimal coding, Streamlit turns Python scripts into interactive web apps, ideal for demonstrating machine learning projects.

7. CSV (Comma-Separated Values)

CSV format is used for saving output data such as path length, number of nodes explored, and computation time. It allows for easy review, analysis, and sharing of results.

V. PROPOSED WORK

The aim of this project is to develop and evaluate a comparative framework between the traditional A* algorithm and a Neural A* algorithm that utilizes a learned heuristic. The proposed work is structured into five comprehensive phases:

1. **Development of a Dynamic Maze Environment**

To begin with, a flexible grid-based maze environment will be designed to simulate various pathfinding challenges. Users will have control over the size of the grid, the density of obstacles, and the start and end points. The environment will ensure that all generated mazes are solvable by checking for connectivity between the start and goal. This phase is critical as it creates a consistent and realistic testing ground for evaluating the algorithms under various scenarios.

2.**Implementationof A-Star and Neural A-Star Algorithms**

The next phase involves implementing two versions of the pathfinding algorithm:

A-Star Algorithm: The traditional A-Star search will use a priority queue (min-heap) to explore nodes based on the cost function f(n) = g(n) + h(n), where g(n) is the cost from the start to node n, and h(n) is the heuristic (estimated cost from n to the goal). The Euclidean distance will be used as the heuristic, and movement will be allowed in four directions.

Neural A-Star Algorithm: This version will use a Convolutional Neural Network (CNN) to learn the heuristic function. The model will be trained using supervised learning, where it imitates the decisions of the A* algorithm on simple training mazes. Once trained, the neural network will predict heuristic values in unseen environments, potentially allowing faster pathfinding with fewer node expansions.

3. **User Interface Development Using Streamlit**

To make the project interactive and visually demonstrative, a Streamlit-based frontend will be developed. This interface will allowDynamic selection between A* and Neural A* algorithmsReal-time visualization of the search process and final pathModification of maze size and obstacle probabilityA feature to download performance metrics and paths in CSV formatStreamlit provides a simple and responsive framework to turn Python scripts into deployable web applications, making the user experience more engaging.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, May 2025
Impact Factor: 7.67

4. **Evaluation and Visualization of Performance Metrics**

This phase involves rigorous testing of both algorithms across various maze configurations. The key performance metrics to be collected and analysed are

Path Length: Measures the efficiency of the final route.

Computation Time: Total time taken by the algorithm to find the path.

Number of Nodes Expanded: Indicates the algorithm's search efficiency.

Visualizations will be created using Matplotlib,including

Side-by-side path overlays for A* and Neural A*Bar charts comparing all performance metricsHeatmaps showing node expansions (optional)These visualizations will help in highlighting differences in algorithmic behavior and efficiency.

5. **Documentation and Analysis**

The final step of the project involves compiling all results and findings into a structured format. This includesA detailed analysis of performance trendsObservations on the strengths and weaknesses of each algorithmDiscussion on where the learned heuristic improves performance and where it may lagSuggestions for improving neural heuristics in future workAll insights and results will be documented to support the research objective of understanding whether machine-learned heuristics can realistically outperform traditional heuristics in pathfinding tasks.

VI. CONCLUSION

This Projectset out to compare the classical A*pathfindingalgorithm with the more recent Neural A* variant that integrates deep learning techniques. The goal was to investigate whether a learned heuristic, powered by a convolutional neural network (CNN), could improve the efficiency of pathfinding in grid-based environments without compromising accuracy. Through careful experimentation, implementation, and analysis, the project provided valuable insights into the strengths and trade-offs of both approaches.

The A* algorithm, being well-established and deterministic, consistently delivered optimal or near-optimal paths across various mconfigurations. It expanded nodes logically based on a predefined heuristic, typically Euclidean or Manhattan distance. However, as map complexity increased, A* required expanding a large number of nodes, leading to higher computational costs. This limitation sparked interest in machine learning-based enhancements, giving rise to the Neural A* approach.

Neural A* attempts to reduce the number of node expansions by learning a heuristic function through a neural network trained on environment data. The trained model demonstrated promising results, particularly in complex or previously unseen environments. It was able to generalize well to new scenarios, often expanding fewer nodes than the traditional A*, while maintaining similar path quality. However, it also introduced challenges such as longer training times and occasional suboptimal paths due to imperfect learning.

Performance metrics such as path length, computation time, and number of nodes expanded were used to compare both algorithms. Visualizations and charts clearly illustrated that while A* maintains consistency, Neural A* offers significant reductions in computational effort in many cases. The Neural A* model is particularly valuable in real-time applications or large-scale environments where traditional heuristics become computationally expensive.

In conclusion, this project highlights the potential of combining classical search methods with machine learning to improve algorithmic efficiency. While A* remains a robust and reliable method, Neural A* opens the door to smarter, more adaptive solutions in dynamic or complex environments. With further enhancements in training quality and network architecture, Neural A* could become a standard in future pathfinding systems. This project lays the foundation for more advanced research into hybrid search-learning systems.

VII. FUTURE SCOPE

While this project successfully demonstrated the comparative performance of A* and Neural A* algorithms, several opportunities remain for further exploration and enhancement. One key area for future work is the improvement of the neural heuristic function. The current CNN model, while effective, can be further optimized using more advanced architectures like attention-based models or transformers, which may offer better generalization and precision in dynamic environments.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 7, May 2025
Impact Factor: 7.67

Another promising direction is the integration of reinforcement learning techniques. Instead of training the neural network in a supervised manner using precomputed paths, reinforcement learning could allow the model to learn heuristics by interacting directly with the environment. This could result in more adaptive and intelligent pathfinding agents, particularly in real-time or stochastic environments where static training data is insufficient.

Additionally, future iterations of this project could focus on scalability and 3D navigation, extending the system from 2D grid-based maps to complex 3D terrains or real-world robotics simulations. Such enhancements would test the robustness of the neural heuristic under more demanding spatial and computational constraints, and would be highly relevant for autonomous vehicles, drones, or robotic exploration.

There is also room to improve the training efficiency and model portability. Techniques like model compression, quantization, or pruning could be explored to reduce the inference time of the neural network without sacrificing accuracy, making Neural A* more suitable for embedded or low-power systems.

Finally, a valuable future addition would be the implementation of a real-time visualization tool or interactive GUI that allows users to compare paths, tweak model parameters, and visualize node expansions dynamically. This would not only enhance usability but also serve as an educational tool for understanding search algorithms and neural pathfinding.

REFERENCES

[1] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100–107.

[2] Silver, D., & Veness, J. (2010). *Monte-Carlo Planning in Large POMDPs*. Advances in Neural Information Processing Systems (NeurIPS), 23.

[3] Y. Chen, D. Zhou, G. Konidaris, and L. P. Kaelbling, "Learning Heuristic Functions for Search-Based Planning," in *Proceedings of the 3rd Conference on Robot Learning (CoRL)*, vol. 100, pp. 409–420, 2020. [Online]. Available: https://proceedings.mlr.press/v100/chen20b.html

[4] Yang, G., Dai, H., Zhang, H., & Song, L. (2020). *Neural A** Search: Towards a Neural Version of Classical Planning. Advances in Neural Information Processing Systems (NeurIPS), 33, 22305–22316.

[5] Bhardwaj, M., Choudhary, A., & Patra, B. (2021). *Neural A-Star* Search: Learning Heuristics for Graph-Based Pathfinding. arXiv preprint arXiv:2102.11859.

[6] Kolter, J. Z., & Abbeel, P. (2008). *Learning Heuristics for A** Search Using Imitation Learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)* (pp. 529–536). ACM.
https://doi.org/10.1145/1390156.1390224

[7] Streamlit Team. (2023). *Streamlit: Turn data scripts into shareable web apps*. Retrieved from https://streamlit.io

[8] MATPLOTLIB DEVELOPERS. (n.d.). *Matplotlib:-*
Visualization Library. https://matplotlib.org/