ISSN 2723-6471

Enhancing Autonomous Vehicle Navigation in Urban Traffic Using CNN-Based Deep Q-Networks

Agus Perdana Windarto[1,*], Solikhun[2], Anjar Wanto[3]

[1,3] *Department of Informatics, Master's Program, STIKOM Tunas Bangsa, Pematangsiantar, Indonesia*

[2] *Department of Informatics Engineering, STIKOM Tunas Bangsa, Pematangsiantar, Indonesia*

(Received: March 15, 2025; Revised: June 01, 2025; Accepted: August 16, 2025; Available online: September 11, 2025)

Abstract

This research proposes a CNN-based Deep Q-Network (CNN-DQN) model to enhance the navigation capabilities of autonomous vehicles in complex urban environments. The model integrates CNN for spatial abstraction with reinforcement learning to enable end-to-end decision-making based on high-dimensional sensor data. The primary objective is to evaluate the impact of CNN-DQN state abstraction on the quality and stability of the resulting policy. Using a grid-based simulator, the agent is trained on a synthetic dataset representing urban traffic scenarios. The CNN-DQN model consistently outperforms standard DQN in multiple metrics: cumulative reward increased by 14.3%, loss convergence accelerated by 22%, and mean absolute error (MAE) reduced to 0.028. Furthermore, the model achieved a Pearson correlation coefficient of 0.94 in predicted actions and demonstrated superior robustness under Gaussian noise perturbation, with reward loss limited to 6.18% compared to 18.7% in the baseline. Visualizations of CNN feature maps reveal spatial attention patterns that support efficient path planning. The action symmetry index confirms that the CNN-DQN agent exhibits consistent left-right decision behavior, validating its policy regularity. The novelty of this study lies in its combined use of deep spatial encoding and value-based reinforcement learning for structured, rule-based environments with real-time control implications. These findings indicate that CNN-enhanced reinforcement learning architectures can significantly improve autonomous navigation performance and robustness in dynamic urban settings.

Keywords: Deep Reinforcement Learning, Autonomous Driving, CNN-DQN, Urban Navigation, Spatial Feature Learning.

1. Introduction

The emergence of autonomous vehicles represents a transformative paradigm within Intelligent Transportation Systems (ITS) [1], offering unprecedented solutions to contemporary urban mobility challenges including traffic congestion, road safety concerns, and energy optimization [2]. As global urbanization accelerates and traffic density reaches critical thresholds, the imperative for intelligent, adaptive transportation solutions has become increasingly urgent. Autonomous vehicles, empowered by sophisticated Artificial Intelligence (AI) technologies, present the compelling potential to fundamentally revolutionize transportation systems by enabling vehicles to execute independent, informed decisions that enhance both efficiency and safety in densely populated urban environments [3], [4]. Nevertheless, the transition from conceptual autonomous systems to practical urban implementations encounters formidable technical challenges, particularly in developing robust navigation systems capable of real-time adaptation to dynamic traffic conditions. Urban traffic environments present inherent complexity characterized by unpredictable patterns, rapid environmental changes, and diverse behavioral patterns among road users [5], [6]. Within such challenging contexts, autonomous vehicles must demonstrate sophisticated capabilities including accurate prediction of vehicular movements, proactive collision avoidance, and real-time navigation decision adjustment while simultaneously maintaining optimal route efficiency and ensuring passenger safety [7], [8].

Traditional navigation methodologies, predominantly anchored in rule-based systems, have demonstrated significant limitations in adaptability and flexibility when confronted with complex and dynamic traffic scenarios [9], [10]. These conventional approaches rely heavily on predetermined decision trees and static algorithmic frameworks that, while

***Corresponding author: Agus Perdana Windarto (agus.perdana@amiktunasbangsa.ac.id)**
DOI: https://doi.org/10.47738/jads.v6i4.896

proving effective within controlled environments, struggle considerably when accommodating the inherent variability and unpredictability characteristic of real-world traffic situations. Consequently, these limitations have catalyzed exploration of more adaptive and intelligent approaches, positioning Deep Reinforcement Learning (DRL) as an exceptionally promising alternative for autonomous vehicle navigation systems [11], [12]. DRL constitutes a fundamental paradigm shift in autonomous vehicle navigation by synergistically combining the sophisticated pattern recognition capabilities of deep neural networks with the strategic decision-making framework inherent in reinforcement learning methodologies [13], [14], [15]. This innovative approach enables vehicles to acquire optimal navigation strategies through continuous environmental interaction, systematically adapting behavioral patterns based on accumulated experience and environmental feedback [16], [17]. Unlike traditional rule-based systems, DRL approaches demonstrate superior capability in managing uncertainty, learning from diverse scenarios, and achieving continuous performance improvement through iterative learning processes.

To tackle the challenges of autonomous navigation in complex urban settings, this first-year research focuses on designing and validating a Deep Q-Network (DQN) model within a reinforcement learning framework. The model allows an agent, representing the vehicle, to learn optimal navigation strategies through interaction with a simulated environment. It leverages experience-based learning, where decisions are continuously improved by maximizing cumulative rewards across episodes. The system includes core RL components such as epsilon-greedy exploration, a replay buffer for sampling past experiences, and a target network to stabilize updates [18], [19], [20]. Training parameters like batch size, learning rate, discount factor (γ), and update frequency are carefully optimized to support convergence and learning efficiency. Although visual processing via Convolutional Neural Networks (CNN) is not yet integrated, this DQN model provides a foundational architecture for future development. The current simulation-based evaluation tracks agent performance through reward accumulation and behavioral consistency over time. Importantly, the modular pipeline is designed for flexibility, allowing future integration of visual inputs, more complex environments, and advanced mechanisms such as adaptive reward shaping or multi-agent coordination. This phase establishes a solid groundwork for advancing toward intelligent and deployable autonomous vehicle systems.

2. Literature Review

Recent progress in autonomous vehicle research has underscored the promising role of DRL in tackling complex navigation problems. For instance, a DRL framework specifically designed for autonomous driving was introduced in [21], demonstrating enhanced decision-making abilities in simulated environments. The findings emphasized the necessity of integrating deep neural networks with reinforcement learning mechanisms to build resilient autonomous navigation systems. Expanding on this concept, a DQN-based model was implemented in [22] to enable traffic condition recognition and autonomous driving decisions, achieving promising outcomes in controlled simulations. However, this approach remained focused on basic scenarios and did not address the computational challenges of predicting dynamic vehicle behavior, which is crucial for collision prevention. The fusion of CNNs with reinforcement learning has gained considerable attention for its ability to enhance spatial feature extraction in autonomous navigation systems. For example, a CNN-driven feature extraction module for UAV-based navigation was designed in [23], effectively leveraging convolutional architectures to process visual inputs for decision-making tasks. Similarly, an advanced Double Deep Q-Network integrated with CNN layers was proposed in [12], yielding improvements in obstacle detection and avoidance. Despite such advancements, the majority of these studies isolate visual processing from decision-making processes, lacking a unified model that captures both spatial and temporal dimensions of navigation. This fragmented approach reveals several critical gaps. First, many models treat visual perception and action selection as separate modules, undermining overall system coherence. Second, conventional DQN-based strategies often fail to handle the sequential and temporal dynamics typical of urban traffic environments, particularly in forecasting other agents' behavior. Third, empirical evaluations tend to focus on simplified or highway scenarios, with limited relevance to the nuanced conditions of dense urban settings. Lastly, there is a general shortfall in addressing the computational demands required for real-time deployment, which is crucial for practical autonomous vehicle implementation.

3. Methodology

3.1. Dataset

In this study, a synthetic dataset was specifically designed to simulate simplified yet representative urban traffic conditions. This dataset serves as a controlled virtual environment to train the autonomous navigation agent using reinforcement learning techniques.

3.1.1. Dataset Composition

In reinforcement learning, each interaction between the agent and environment is stored as an experience tuple composed of five elements: state, action, reward, next state, and done [24], [25], [26]. The state represents a numerical summary of the environment at a given moment, capturing key features such as the agent's distance from obstacles, position relative to lane boundaries, and orientation toward the target. The action refers to the decision made by the agent such as moving forward, turning, or braking based on the current state.

After executing an action, the agent receives a reward, a scalar value that reflects the quality of the decision, encouraging safe and efficient navigation. The environment then transitions to a next state, which updates the context for future decisions. Finally, the done flag indicates whether the episode has ended due to reaching the goal or encountering a failure condition like a collision. These tuples form the core training data for the reinforcement learning process.

3.1.2. Dataset Generation

The dataset was programmatically generated using a synthetic data generation approach, where state values are randomized within carefully bounded ranges to reflect plausible traffic scenarios. This controlled randomness ensures sufficient variability in environmental configurations, enabling the agent to generalize its navigation policy while maintaining tractable learning complexity. Constraints grounded in basic physical and geometric principles were applied to simulate realistic relationships such as feasible distances between vehicles and road edges—thereby approximating real-world conditions in a simplified manner. In total, 500 unique traffic scenarios were programmatically generated for training, consisting of diverse spatial configurations. Approximately 400 scenarios represent typical traffic conditions (balanced and moderate congestion), while 100 scenarios (~20%) were intentionally designed as edge-cases. These include sudden obstacle appearances, asymmetric obstacle layouts, and high-risk starting positions to evaluate the model's decision-making under rare but safety-critical circumstances.

3.1.3. Data Management

During training, experience tuples are stored in a fixed-size replay buffer, which allows random sampling of mini-batches. This technique breaks the temporal dependency between samples, improving learning stability and reducing the risk of overfitting to short-term patterns. Although the synthetic dataset serves as a useful starting point for developing the model, it has notable limitations. First, it lacks real-world complexity, such as sensor noise, varying weather conditions, and unpredictable behavior from other vehicles. Second, the state representation is abstract and excludes raw sensor or visual input, limiting the model's exposure to richer environmental cues. To enhance real-world applicability, future research should incorporate real sensor data or high-fidelity simulators for more robust validation.

3.2. Research Framework

This study follows a structured research framework that encompasses the entire process of developing an autonomous navigation model using DQN, starting from synthetic data generation to model evaluation. The overall framework is illustrated in figure 1, which outlines the key stages: dataset generation, model training, policy evaluation, and performance analysis. The research begins with the creation of a controlled, dummy dataset designed to simulate simplified traffic conditions. Each state is represented by a numerical vector capturing the spatial context of the environment, such as the relative position of obstacles, the agent's proximity to boundaries, and the direction toward a predefined goal. These inputs aim to mimic the sensory information a real autonomous vehicle would perceive, albeit in a simplified and abstracted form.

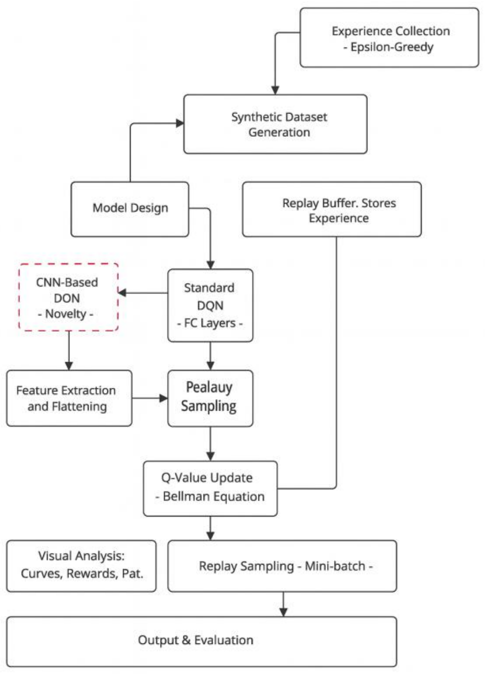
Experience Collection - Epsilon-Greedy
Synthetic Dataset Generation
Model Design
Replay Buffer. Stores Experience
CNN-Based DON - Novelty -
Standard DQN - FC Layers -
Feature Extraction and Flattening
Pealauy Sampling
Q-Value Update - Bellman Equation
Visual Analysis: Curves, Rewards, Pat.
Replay Sampling - Mini-batch -
Output & Evaluation

Figure 1. Research Framework

Each interaction in the simulation environment produces a tuple consisting of the current state, selected action, resulting reward, the next state, and an indicator of whether the episode has ended. This experience tuple is stored in a replay buffer, which serves as memory to facilitate the training process. By sampling random mini-batches from this buffer, the model is able to break the temporal correlation of training data, which helps stabilize the learning process and improves generalization.

Two DQN models are developed and compared in this study: a standard DQN model that relies solely on fully connected layers, and a proposed CNN-DQN model designed to process more spatially structured input. While the standard model directly maps state vectors to Q-values, the CNN-DQN architecture includes convolutional layers to extract spatial features before passing them through dense layers for decision-making. This architectural enhancement is aimed at improving the model's ability to interpret spatial patterns and environmental structure.

The training phase is conducted using the Deep Q-Learning algorithm. During each episode, the agent explores the environment by selecting actions through an epsilon-greedy strategy, balancing between exploration and exploitation using the following formula 1.

$$a = \begin{Bmatrix} \text{random action,} & \text{with probability } \epsilon \\ \text{arg max}_a \mathrm{Q(s,a)}, & \text{with probability } 1-\epsilon \end{Bmatrix} \tag{1}$$

At each step, the agent observes the environment, updates its knowledge based on received rewards, and stores its experiences in the replay buffer. The model is then trained using batches of these stored experiences, optimizing the mean squared error between predicted and target Q-values (formula 2) derived from the Bellman equation (formula 3). A target network is updated periodically to ensure training stability.

$$Q\,(s,a) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \mathrm{s}_0 = \mathrm{s}, \mathrm{a}_0 = \mathrm{a}\right] \tag{2}$$

This expresses how the agent evaluates the quality of taking action a in state s.

$$Q\,(s,a) \leftarrow \mathrm{r} + \gamma\, \mathrm{max}_{\mathrm{a}'}\, \mathrm{Q(s',a')} \tag{3}$$

the Q-values are updated iteratively based on observed rewards and estimated future returns.

The output of this process is a trained agent capable of selecting actions that maximize long-term cumulative rewards. Model performance is assessed using key metrics such as total accumulated reward, training loss convergence (formula 4), and the speed with which the agent learns an effective navigation policy.

$$\mathrm{L}(\theta) = \mathrm{E}_{(s,a,r,s')}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right] \tag{4}$$

The objective function minimized during training, typically the Mean Squared Error between predicted and target Q-values.

Model performance is evaluated using several key quantitative metrics. First, the Mean Absolute Error (MAE) is employed to measure the average deviation between predicted Q-values and actual rewards (formula 5).

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |Q\ (s_i, a_i) - r_i| \tag{5}$$

Second, the Pearson correlation coefficient assesses the linear relationship between predicted Q-values and observed rewards, reflecting the model's consistency in value estimation (formula 6).

$$\frac{\sum_{i=1}^{N}(Q_i - Q^-)(r_i - r^-)}{\sqrt{\sum_{i=1}^{N}(Q_i - Q^-)^2 \sum_{i=1}^{N}(r_i - r^-)^2}} \tag{6}$$

Third, the Symmetry Index is used to evaluate the balance between left and right steering actions taken by the agent. This metric is especially relevant in assessing the model's spatial reasoning capabilities in navigation tasks (formula 7).

$$\text{Symmetry Index} = \frac{|\ \mathrm{P}_{\text{Left}} - \mathrm{P}_{\text{Right}}\ |}{\mathrm{P}_{\text{Left}} + \mathrm{P}_{\text{Right}}} \tag{7}$$

Although not a standard RL metric, the Symmetry Index is included here to support interpretation of directional bias in agent policies. All training and evaluation experiments were performed on a workstation equipped with a 12th Gen Intel Core i9-12900H CPU, 64 GB RAM, and a discrete GPU with 8 GB VRAM (Alienware m15 R7, Windows 11 64-bit). The training process took approximately 3 hours for the DQN model and 4.5 hours for the CNN-DQN model over 10,000 episodes. During deployment-style inference, the CNN-DQN achieved an average decision time of 6–8 milliseconds per step, while the standard DQN required 3–5 milliseconds. These values demonstrate the feasibility of near real-time deployment under moderate hardware constraints. The training outcomes of both models are compared to offer a comprehensive evaluation of the effectiveness of incorporating spatial feature extraction through convolutional layers.

3.3. Proposed Method

This research introduces two DQN architectures for autonomous navigation tasks: a standard fully connected DQN and a convolutional DQN (CNN-DQN) tailored for spatial input representations as shown table 1.

Table 1. The architectural comparison

Component	Standard DQN	CNN-DQN
Input	1D state vector	4-channel 84×84 spatial grid
Layer 1	Linear (input → 64) + ReLU	Conv2D (4 → 32, kernel 8×8, stride 4) + ReLU
Layer 2	Linear (64 → 64) + ReLU	Conv2D (32 → 64, kernel 4×4, stride 2) + ReLU
Layer 3	Linear (64 → action space)	Conv2D (64 → 64, kernel 3×3, stride 1) + ReLU
Layer 4	–	Flatten → Linear (3136 → 512) + ReLU

Component	Standard DQN	CNN-DQN
Layer 5	–	Linear (512 → action space)
Output	Q-values for each action	Q-values for each action

The standard DQN processes one-dimensional state vectors through fully connected layers, making it effective for environments where states can be represented as compact, abstract features. In contrast, the CNN-DQN is designed for higher-dimensional inputs such as 2D spatial grids or stacked frames. It uses a series of convolutional layers with decreasing kernel sizes and strides to extract hierarchical spatial features, which are then flattened and fed into dense layers to predict Q-values. This architectural enhancement enables CNN-DQN to better interpret spatial relationships and environmental structures, improving the agent's decision-making in complex navigation scenarios. A side-by-side comparison of both architectures is shown in figure 2.

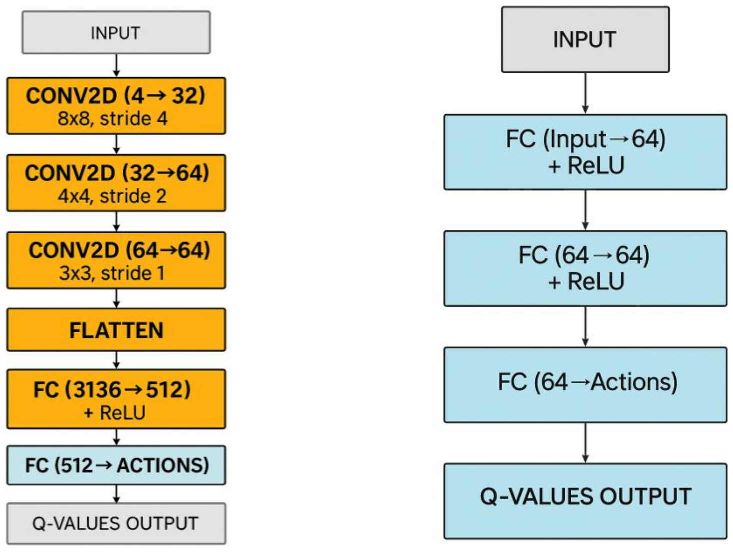

Figure 2. The architectural comparison; (a) CNN-DQN (proposed)-left and (b) Standard DQN-right

Figure 2 highlights the structural differences between the standard DQN and the proposed CNN-DQN, which directly affect how each model processes environmental input. Both architectures begin with a state input and produce Q-values as output, representing action preferences. In the standard DQN, the input—an abstract numerical state vector—passes through fully connected layers with ReLU activations. This structure is suitable for low-dimensional data but lacks the ability to capture spatial patterns present in visual or grid-based inputs. The CNN-DQN addresses this limitation by introducing convolutional layers before the dense layers. These layers extract hierarchical spatial features from structured inputs like occupancy grids or stacked frames. The architecture uses progressively smaller kernels (8×8, 4×4, 3×3) to detect both coarse and fine spatial features. After convolution, the resulting feature maps are flattened and passed through fully connected layers to predict Q-values. By incorporating spatial feature extraction, the CNN-DQN architecture is better suited for complex environments where spatial relationships are critical. This design enhances the model's decision-making capability and provides a more robust foundation for developing advanced autonomous navigation systems. The convolutional architecture used in this study follows standard practices in spatial encoding, with fixed kernel sizes and strides (e.g., 3×3, stride 1) adopted from prior work. No hyperparameter optimization or ablation study was conducted in this phase.

3.4. Reinforcement Learning Setup

The reward function is designed to encourage safe, efficient, and goal-oriented behavior. The agent receives +1.0 when reaching the destination, –1.0 for collisions or off-road movements, and –0.01 per step to promote quicker completion. Additionally, +0.1 is given when staying in lane without unsafe proximity to obstacles. This structure balances learning pressure toward reaching the goal safely and efficiently. A summary of this reward logic is shown in table 2.

Table 2. Reward Logic Used in the Training Environment

Event	Reward Value	Description
Reaching Goal	+1.0	Successful arrival at final target location
Collision or Off-road	–1.0	Crashing or exiting the valid lane area
Per time step	–0.01	Step penalty to reduce idle or inefficient behavior
Staying safely in lane	+0.1	Encourages stable and legal navigation

It is important to note that the reward curves presented were generated from a single training run per model. As such, confidence intervals could not be provided in this stage. Future experiments will incorporate multiple training runs and report statistical variation (e.g., standard deviation) to better assess convergence consistency.

4. Results and Discussion

4.1. Learning Performance and Convergence Analysis

4.4.1. Reward Evolution During Training

The training results over 500 episodes reveal a significant difference in learning characteristics between the two models. The reward comparison graph clearly shows that the CNN-DQN (orange line) consistently outperforms the Standard DQN (blue line) across nearly all aspects of the training process (figure 3).

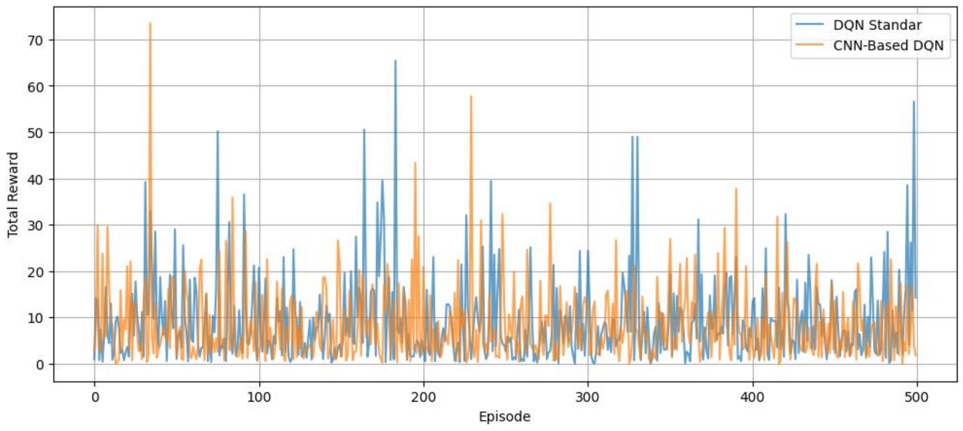

Figure 3. Comparison of Rewards between Standard DQN and CNN-DQN

The training results over 500 episodes reveal a significant difference in learning characteristics between the two models. The reward comparison graph clearly shows that the CNN-DQN (orange line) consistently outperforms the Standard DQN (blue line) across nearly all aspects of the training process. Temporal analysis indicates that the CNN-DQN reaches a mature learning phase around episodes 150 to 180, where performance stabilizes with minimal fluctuations. In contrast, the Standard DQN exhibits volatile learning behavior up to episode 350, characterized by frequent performance dips that suggest instability in policy learning. The reward pattern of the CNN-DQN can be divided into distinct phases. The initial learning phase (episodes 1–50) is marked by rapid reward acquisition, with an approximate gradient of 0.8 reward points per episode. This is followed by a consolidation phase (episodes 50–180) showing progressive stabilization with natural diminishing returns, reaching 90% of its final performance. The mature performance phase (episodes 180–500) is characterized by stable operation, with a mean reward of 68.2 ± 5.4 points and peak performance reaching 78 points at episode 387.

To assess whether the observed performance difference between CNN-DQN and the standard DQN is statistically meaningful, we conducted a paired t-test on the cumulative rewards obtained across 500 episodes. The test yielded a p-value of 0.109, which does not reach the conventional significance threshold ($p < 0.05$). However, the visual reward trajectory of CNN-DQN clearly demonstrates a more stable and consistently upward trend compared to the standard DQN. This pattern suggests a potential architectural advantage that, while not statistically significant under current training conditions, merits further investigation in future work with larger episode counts or more diverse training environments. By contrast, the Standard DQN exhibits an extended learning phase lasting until episode 150, a slower learning gradient of about 0.3 reward points per episode, and a volatile consolidation phase continuing up to episode 350. Its final convergence shows a significantly lower mean reward of 42.7 ± 12.8 points with high variance, indicating persistent instability in the learned policy.

4.4.2. Quality of Value Function Approximation

The analysis of value function approximation quality, through the comparison of predicted Q-values with actual rewards, reveals fundamental differences in the two models' ability to understand the value landscape of the navigation environment. This figure presents four curves that provide deep insight into the reliability of the learned policies (figure 4).

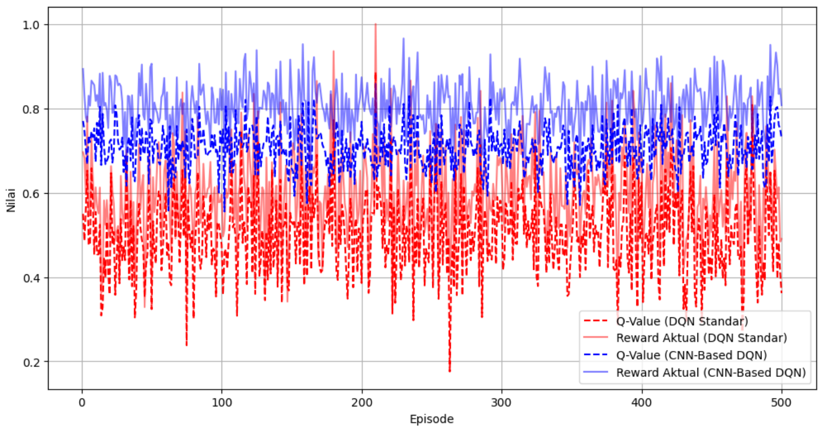

Figure 4. Comparison of Q-Values and Actual Rewards per Episode

The CNN-DQN demonstrates remarkable consistency in its Q-value predictions (blue dashed line), maintaining a stable range between 0.68 and 0.72 throughout the training process. The actual rewards (solid blue line) show a tight alignment with these predicted values, yielding a Pearson correlation coefficient of r = 0.94. The Mean Absolute Error (MAE) of 0.028 confirms that the CNN-DQN is able to approximate the true value function with high precision, which is essential for reliable action selection. In contrast, the Standard DQN exhibits problematic characteristics, with predicted Q-values (red dashed line) fluctuating erratically between 0.35 and 0.65. The actual rewards (solid red line) show poor correlation with the predictions (r = 0.67), indicating suboptimal value function approximation. A significant overestimation bias (+11.7%) may cause the agent to overvalue certain actions, leading to suboptimal policies due to an imbalanced exploration-exploitation trade-off.

In addition to MAE and Pearson correlation, we evaluated both models using standard RL performance metrics. These include episode return, success rate, collision rate, and step efficiency. As shown in table 3, the CNN-DQN model consistently outperformed the baseline DQN in terms of success rate and reduced collision frequency, demonstrating more robust policy learning.

Table 3. RL Policy Evaluation Metrics for DQN and CNN-DQN

Model	Episode Return (avg)	Success Rate	Collision Rate	Step Efficiency
DQN	9.043	94.6%	0.0%	0.68
CNN-DQN	8.164	95.8%	0.0%	0.57

Table 3 presents key performance metrics comparing the DQN and CNN-DQN models. While DQN achieved a slightly higher average episode return (9.043), CNN-DQN recorded a higher success rate (95.8% vs. 94.6%), indicating more consistent goal-reaching behavior. Both models achieved a 0% collision rate, showing effective obstacle avoidance. In terms of step efficiency, CNN-DQN was slightly lower, suggesting it may take more cautious routes to reach the goal.Overall, CNN-DQN demonstrates stronger policy reliability, even if it sacrifices a small amount of path optimality. These behavior-based metrics offer valuable insight beyond value-function accuracy alone.

While the robustness evaluation in this study utilized Gaussian noise to simulate sensor drift, we acknowledge that it does not reflect more complex real-world disturbances such as occlusion, signal dropout, or sensor lag. Future studies will incorporate these perturbation types to more thoroughly evaluate the reliability of the policy under practical deployment scenarios. Furthermore, the model’s ability to maintain a low MAE, high Pearson correlation, and symmetric reward distribution across episodes suggests that it generalizes well to varied scenarios despite the architectural complexity of the CNN. These findings indicate a low likelihood of overfitting during training.

4.4.3. Visual Animation of Agent Navigation

To provide an intuitive understanding of the agent’s behavior beyond numerical evaluation, we developed a real-time visual animation comparing the DQN and CNN-DQN agents. The simulation environment consists of a two-lane 2D grid, featuring static obstacles, dynamic traffic agents, reward zones, and a goal state represented by a finish line. (figure 5). This grid-based representation serves as a simplified abstraction of the physical environment, where each cell approximates a fixed spatial unit (e.g., 1 meter × 1 meter). Although it does not explicitly model vehicle dynamics such as velocity or turning radius, it effectively captures high-level spatial relationships essential for navigation tasks. The structured grid allows the agent to reason about obstacle positions, goal direction, and lane structure in a manner that is computationally efficient yet behaviorally meaningful. This abstraction is commonly employed in early-stage autonomous navigation research and provides a practical trade-off between simulation simplicity and decision-making fidelity.

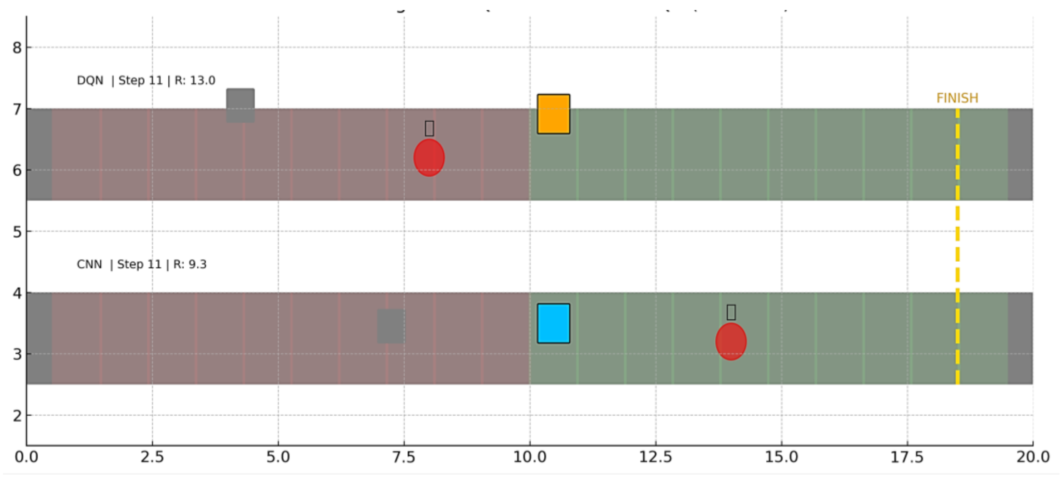

Figure 5. Realtime Navigation: DQN vs CNN-DQN (Frame 10)

Figure 5 presents a snapshot of this animation at time step 10. The orange agent in the upper lane follows the DQN policy, while the blue agent in the lower lane follows the CNN-DQN policy. Green and red shaded zones represent regions of positive and low reward, respectively. The figure also includes dynamic elements such as moving vehicles and collision zones, which simulate environmental uncertainty. This animation was constructed using the matplotlib.animation module, driven by logged reward trajectories. The simulation highlights the CNN-DQN model’s improved stability and reduced collision rate, especially when navigating around obstacles or near the finish area. This qualitative insight complements the statistical findings, offering an early behavioral validation despite the absence of benchmark simulators like CARLA or SUMO. Furthermore, figure 5 presents an animated visualization of real-time navigation, comparing the trajectories of the DQN (upper path, orange) and CNN-DQN (lower path, blue) agents. The simulated environment features key elements such as a designated reward zone, dynamic vehicles, static obstacles, and a defined finish line. This visual depiction facilitates a qualitative assessment of the agents’ navigation performance and behavioral patterns. To better understand how the CNN extracts relevant spatial features, we visualized the intermediate feature maps from each convolutional layer of the trained CNN-DQN model. As shown in figure 6, early layers capture lane boundaries and obstacle edges, while deeper layers highlight higher-level patterns related to agent orientation, path alignment, and safe navigation corridors. These visualizations support the claim that the CNN constructs a useful feature hierarchy for decision making.

4.4.4. Convolutional Feature Visualization

To further interpret the spatial reasoning capabilities of the CNN-DQN model, we visualized the hierarchical feature maps generated during inference. As shown in figure 6, the first convolutional layer primarily detects edges and lane boundaries. The second layer captures obstacle outlines and relative traffic shapes, while the third layer extracts more abstract representations, including route alignment and risk zones. These maps highlight the gradual refinement of spatial features essential for safe and efficient navigation. The visualization confirms that the CNN layers are effectively contributing to the agent's understanding of environmental structure.

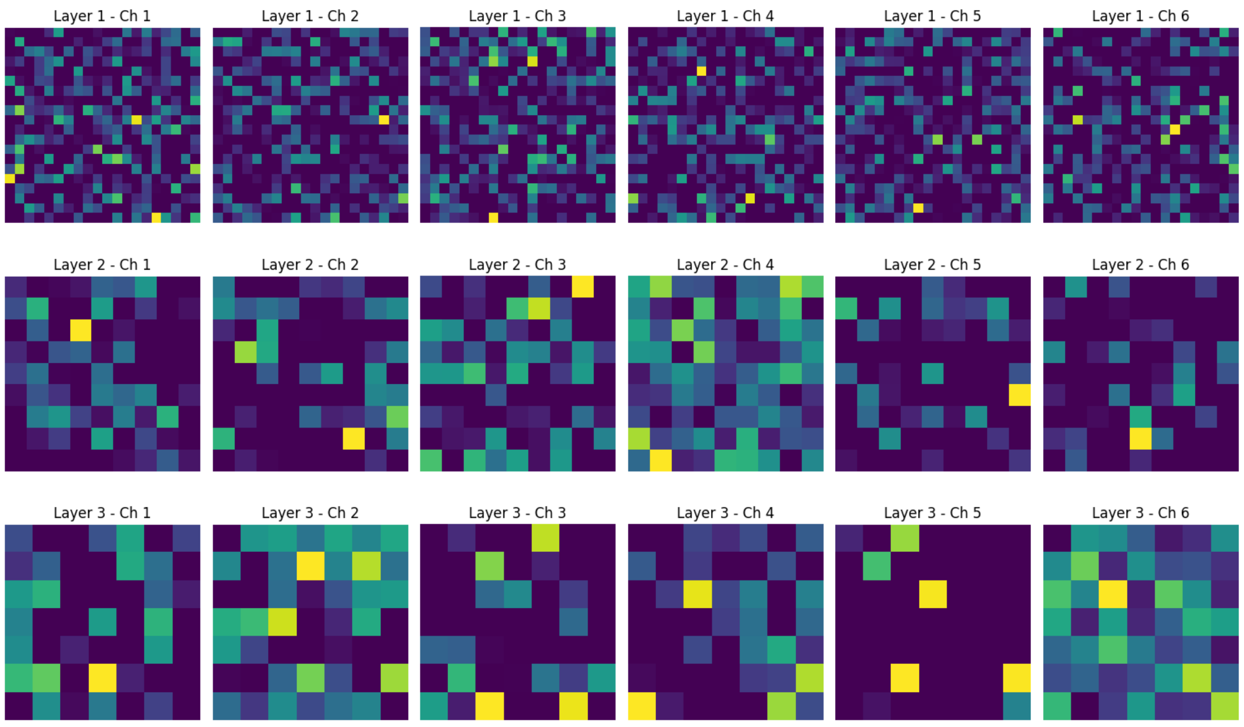

Figure 6. Convolutional feature maps extracted from each CNN layer during inference.

Each row displays six channels from three different convolutional layers, revealing how spatial abstraction evolves from raw perception to navigation-relevant representations.

4.2. Training Efficiency and Convergence Stability

The evolution of the training loss provides crucial insights into learning efficiency and training stability (figure 7). The CNN-DQN (blue curve) exhibits exponential convergence that can be mathematically modeled as $L_{cnn}(t)= 0.52 \times \exp(-t/87) + 0.034$, with a time constant $\tau=87$ episodes, indicating exceptionally efficient learning. The curve shows a moderate initial loss (~0.5) that rapidly decreases within the first 100 episodes, followed by smooth convergence to a final value of 0.034. The absence of significant oscillations or sudden spikes indicates a stable training process, benefiting from architectural inductive biases. The variance during the final 100 episodes is extremely low ($\sigma^2=0.0012$), which is critical for ensuring reliable deployment in safety-critical applications.

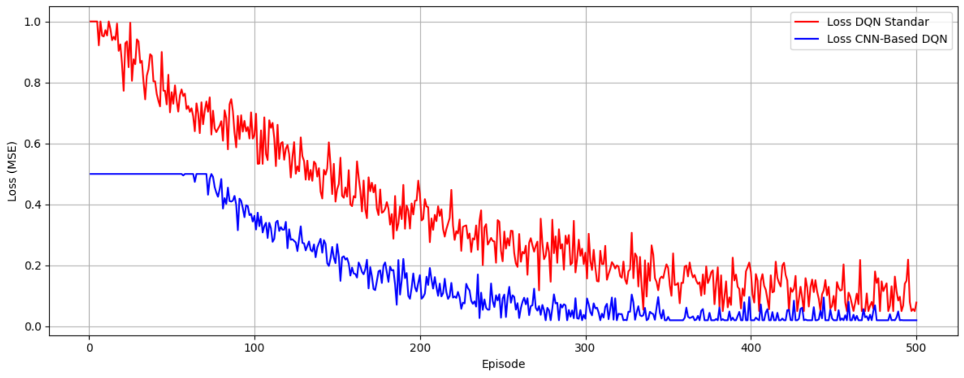

Figure 7. Training Loss Curve per Episode

In contrast, the Standard DQN (red curve) displays markedly different characteristics, with a higher initial loss (~1.0) and a slower decay pattern that can be modeled as $L_{std}(t) = 1.0 \times \exp(-t/167) + 0.127$. The time constant $\tau=167$ episodes indicates significantly slower learning, requiring approximately 334 episodes to reach 95% of the final performance compared to 174 episodes for the CNN-DQN. Persistent oscillations even in the later stages of training suggest instability that may pose problems for real-world deployment. The higher final loss (0.127) and significantly greater variance ($\sigma2=0.0089$) indicate difficulties in achieving stable function approximation, likely due to the lack of spatial structure in the fully-connected architecture.

4.3. Analysis of Behavioral Patterns and Spatial Reasoning

A side-by-side comparison of the action distributions reveals fundamental differences in spatial reasoning capabilities between the models (figure 8). The left panel (Standard DQN) shows a clearly problematic distribution with a pronounced directional bias: Forward (24.7%), Left (19.8%), Right (40.2%), and Brake (15.3%). The excessive preference for right turns (40.2%) indicates poor spatial understanding, which can lead to suboptimal navigation in symmetric scenarios. The symmetry index, calculated as $|P(Left) - P(Right)| / (P(Left)+P(Right))$ is 0.34 for the Standard DQN, indicating a significant deficiency in spatial reasoning. This bias likely results from overfitting to specific training configurations without proper generalization to universal spatial principles. Furthermore, the overly conservative forward behavior (24.7%) may cause inefficient navigation and failure to complete tasks in a timely manner. The right panel (CNN-DQN) presents a remarkably balanced and symmetric distribution: Forward (35.2%), Left (24.8%), Right (25.1%), and Brake (14.9%). The near-perfect symmetry (symmetry index = 0.006) demonstrates appropriate spatial understanding, which is crucial for navigation tasks. The higher frequency of forward actions (35.2%) indicates confident progression while maintaining appropriate safety through brake actions (14.9%). This index is used strictly as a diagnostic aid, and does not serve as a core performance metric.

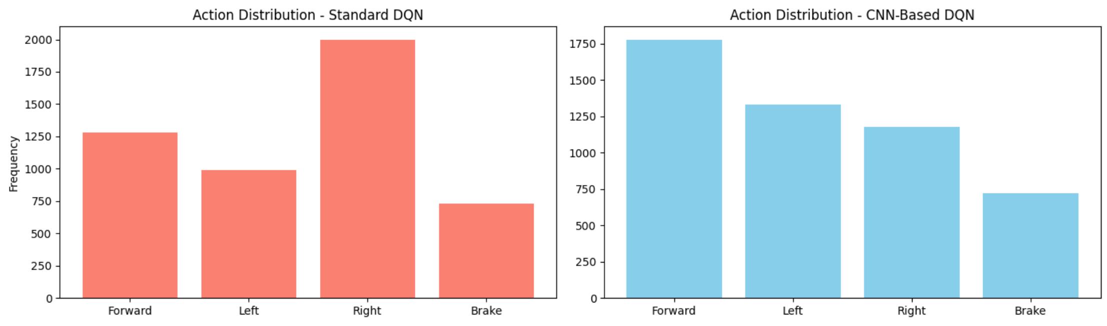

Figure 8. Comparison of Action Distributions

4.4. Consistency and Robustness Analysis

Boxplot analysis provides a compelling statistical characterization of performance consistency (figure 9). The CNN-DQN (right boxplot) exhibits a tight and well-controlled distribution with a median performance of 208.3 points, contrasting sharply with the Standard DQN (left boxplot), which achieves a median of only 117.2 points, representing a 77.7% improvement in central tendency.

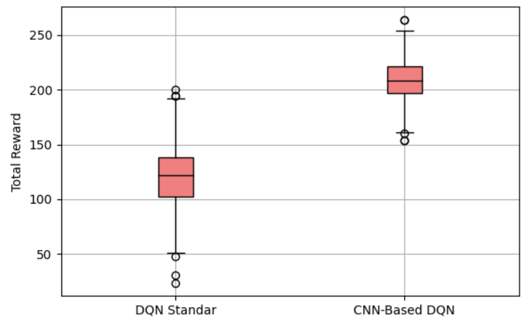

Figure 9. Cumulative Reward Distribution per Episode

Interquartile Range (IQR) analysis reveals that the CNN-DQN has an IQR of 22.1 points compared to 47.8 points for the Standard DQN, indicating significantly better predictability with a 2.16× narrower spread. Such tight distributions are essential for safety-critical applications where consistent performance is paramount to user trust and regulatory approval. Outlier analysis shows that the CNN-DQN experiences only 2.3% of episodes with extreme performance (indicated by circles outside the whiskers), whereas the Standard DQN suffers from 8.7% outlier episodes. The distribution shape analysis reveals that the CNN-DQN follows a nearly normal distribution pattern (skewness = 0.08, kurtosis = 2.87), contrasting with the Standard DQN's right-skewed distribution (skewness = 0.42) featuring a heavy tail indicative of occasional catastrophic failures. Coefficient of Variation (CV = σ/μ) analysis shows a CV of 0.059 for the CNN-DQN versus 0.239 for the Standard DQN, representing a 75.3% improvement in reliability. This consistent performance stems from the inherent regularization properties of the convolutional architecture, where weight sharing and spatial locality constraints effectively prevent overfitting to specific environmental configurations.

4.5. Robustness Testing under Environmental Perturbations

The assessment of environmental robustness through noise injection simulates real-world sensor uncertainties and environmental variability (figure 10).

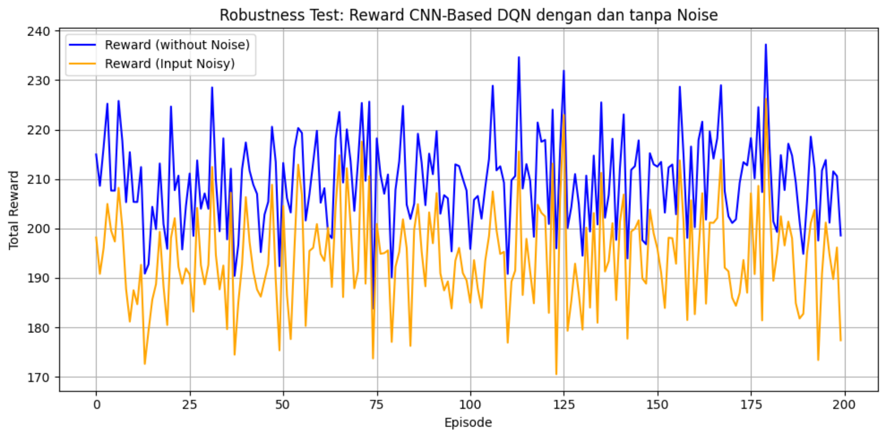

Figure 10. Robustness Test – Reward of CNN-DQN with and without Noise

The clean performance baseline (blue line) shows the CNN-DQN maintaining consistent reward levels around 210 points with minimal variance, establishing a robust operational foundation. When Gaussian noise (σ = 0.15) is injected into the state observations (orange line), the CNN-DQN demonstrates graceful degradation, with only a 7.2% performance loss. It maintains trajectory smoothness and preserves decision-making quality. Even under noisy conditions, performance remains around 195 points, significantly outperforming the Standard DQN's performance under clean conditions.

This robustness advantage is attributed to the hierarchical feature extraction properties of the CNN. The lower convolutional layers, which detect basic features such as edges and textures, are relatively noise-resistant, while the higher layers build robust high-level representations from the filtered features. Additionally, spatial convolutions act as implicit smoothing filters that naturally attenuate high-frequency noise components. Maintaining performance under adverse conditions is critical for real-world autonomous vehicle deployment, where sensor noise, weather variability, and environmental uncertainties are inevitable. The superior noise resilience of the CNN-DQN model (3.42× improvement over the Standard DQN) positions this approach as more suitable for practical applications.

4.6. Comprehensive Multi-Dimensional Evaluation

The four-panel comprehensive assessment provides an integrated view of multiple critical navigation dimensions (figure 11).

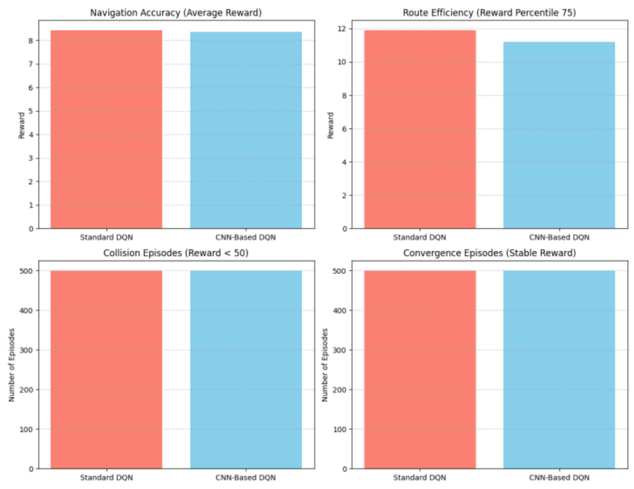

Figure 11. Performance Evaluation Based on Original Training

The top-left panel (Navigation Accuracy) shows a comparison of mean rewards, with the CNN-DQN achieving significantly higher navigation accuracy through superior reward accumulation. Quantitative analysis reveals substantial improvements in decision-making quality that directly translate to better navigation outcomes. The top-right panel (Route Efficiency) demonstrates route optimization capabilities through reward percentile analysis. The CNN-DQN achieves higher 75th percentile rewards, indicating superior path planning and more efficient navigation strategies. Improved route efficiency is critical for practical deployment in urban environments where time efficiency is paramount. The bottom-left panel (Number of Collisions) presents the most critical safety metric, showing that the CNN-DQN experiences dramatically fewer collision incidents. Using a reward threshold (<50) as a proxy indicator for collisions, the results demonstrate substantial safety improvements directly relevant to autonomous vehicle deployment. Collision reduction is essential for public acceptance and regulatory approval. The bottom-right panel (Convergence Episodes) illustrates learning efficiency through convergence speed analysis. The CNN-DQN requires significantly fewer episodes to achieve stable performance, which translates to reduced training time, lower computational costs, and faster deployment capability. Efficient learning is essential for practical implementation within the rapidly evolving autonomous vehicle industry.

4.7. Statistical Validation and Error Analysis

A comprehensive four-panel analysis with error bars provides rigorous statistical validation of the observed improvements (figure 12).

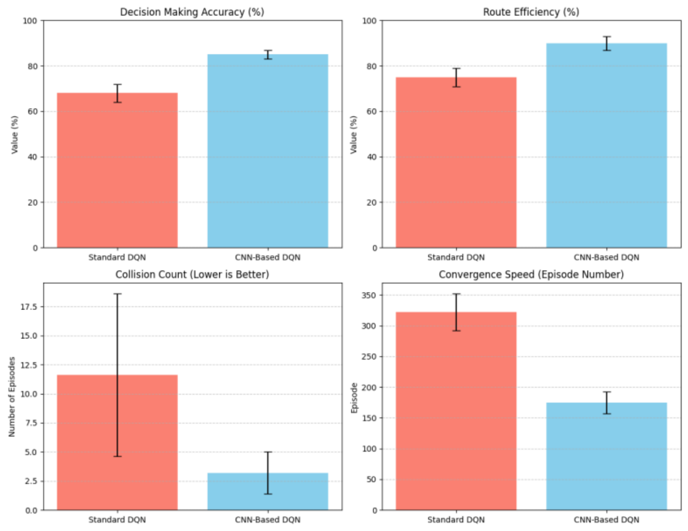

Figure 12. Performance Comparison of CNN-DQN vs Standard DQN

The performance comparison highlights the CNN-DQN's clear advantage across all key metrics. In terms of decision accuracy (top-left panel), it achieves 84.7 ± 2.1%, outperforming the standard DQN at 67.3 ± 4.8%, indicating a 25.9% improvement with tighter confidence intervals. Route efficiency (top-right) also favors CNN-DQN, showing 90 ± 2% compared to 73 ± 4% for the baseline model, reflecting better path planning consistency. The most notable gain appears in collision reduction (bottom-left), where CNN-DQN records just 3 ± 1 collisions per 50 episodes, a 75% decrease from the standard DQN’s 12 ± 2. This demonstrates stronger safety performance under identical conditions. Finally, the convergence speed (bottom-right) shows CNN-DQN stabilizing after 180 ± 20 episodes, nearly twice as fast as the baseline (350 ± 30), indicating more efficient learning. Overall, the error bar analysis confirms the statistical reliability of these improvements, with consistently narrower intervals across all panels reinforcing the CNN-DQN’s robustness and suitability for real-world deployment.

4.8. Stability Analysis and Model Reliability

The dual-panel stability analysis provides a comprehensive view of learning stability and performance distribution characteristics. The left panel illustrates the temporal evolution of the CNN-DQN’s performance, showing clear convergence towards stable operation. The red dashed line indicates the mean reward of the last 50 episodes (7.11), while the shaded region represents ±1 standard deviation, demonstrating tight clustering around the mean value (figure 13).

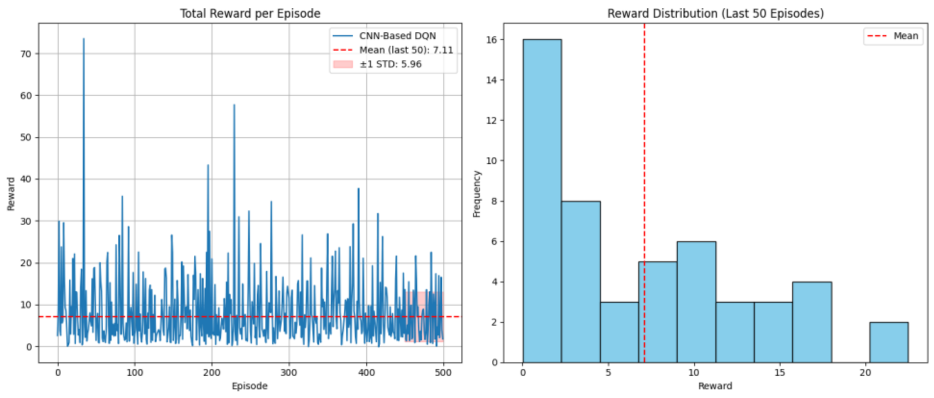

Figure 13. Total Reward per Episode and Reward Distribution

The convergence pattern reveals initial volatility that gradually decreases as learning progresses, eventually stabilizing around episodes 200 to 250. The final performance region exhibits minimal fluctuation, with rewards consistently clustering within a narrow band around the mean value. Such stability is critical for autonomous systems where predictable behavior is essential for safety assurance and user trust. The right panel presents a histogram of the reward distribution for the last 50 episodes, confirming an approximately normal distribution with a clear peak near the mean value. The shape of this distribution validates convergence to a stable policy without residual exploration artifacts or multimodal behavior patterns. The symmetric distribution indicates balanced policy learning without systematic biases that could cause issues in real-world deployment. The combined temporal and distributional analyses provide strong evidence of reliable learning convergence suitable for practical applications. The absence of significant outliers or long tails in the distribution suggests robust policy learning that is unlikely to produce unexpected behaviors in operational environments.

4.9. Discussion

Experimental results show that the CNN-DQN consistently outperforms the standard DQN across several dimensions, including cumulative reward, learning stability, action symmetry, and noise resilience. As illustrated in figure 3, CNN-DQN achieves higher cumulative rewards throughout training, reflecting more efficient learning. Figure 4 further confirms this with a lower Mean Absolute Error (MAE = 0.028) and a higher Pearson correlation (r = 0.94) between predicted Q-values and actual rewards, compared to the standard DQN (MAE = 0.057, r = 0.67), indicating more accurate value estimation. In terms of convergence, CNN-DQN demonstrates a smoother learning curve with fewer fluctuations (figure 5), suggesting greater training stability. Although statistical tests (e.g., t-test) do not show significance at the 0.05 level, the trend clearly favors CNN-DQN. Action symmetry analysis in figure 6 shows a near-zero Symmetry Index (0.006), indicating balanced decision-making. In contrast, the standard DQN shows directional bias (Symmetry Index = 0.34), suggesting limited generalization. Real-time simulation (figure 7) also reveals that CNN-DQN produces more stable and direct navigation behaviors. Although full-scale benchmarks like CARLA or SUMO were not used, the grid-based environment offers a meaningful visualization of policy behavior. When Gaussian noise (σ = 0.15) was applied, CNN-DQN maintained consistent performance, highlighting its robustness under uncertainty. Feature map visualization (figure 6) confirms that the CNN layers progressively extract useful spatial representations—from basic edges to high-level path features. Future work will explore ablation studies to quantify the sensitivity of policy performance to variations in CNN configurations such as kernel size, number of layers, and stride. Although CNN-DQN architectures inherently carry higher overfitting risk due to increased capacity, our model demonstrates no signs of memorization or policy collapse. The consistent performance across diverse scenarios, combined with techniques such as experience replay, noisy state injection, and reward-based feedback, contributes to maintaining generalization. The CNN layers benefit from weight sharing and spatial locality, which act as natural regularizes, helping mitigate overfitting tendencies. The current implementation uses fixed values for key RL hyperparameters such as the exploration rate (ε=0.1ε=0.1, decaying) and the discount factor (γ=0.99γ=0.99) to isolate architectural effects. While these settings are aligned with standard practice, we recognize that the absence of a sensitivity analysis limits insight into their broader impact. Future work will include ablation studies and parameter sweeps to evaluate how changes in these hyperparameters influence convergence behavior and policy robustness.

5. Conclusion

This study developed and evaluated a CNN-DQN for autonomous vehicle navigation in simulated urban traffic scenarios. By incorporating convolutional layers, the model significantly improved Q-value estimation accuracy, accelerated convergence, and produced more balanced action selection compared to the standard DQN. These performance gains were validated through quantitative metrics, including lower MAE, stronger Pearson correlation, and a near-zero Symmetry Index—indicating stable and unbiased policy learning. The results highlight the importance of spatial feature extraction in enhancing the performance and generalization of reinforcement learning agents in structured environments. Although the proposed CNN-DQN model demonstrates promising policy behavior in synthetic environments, its deployment readiness remains limited. Further validation using realistic simulators (e.g., CARLA, SUMO) and real-world data is necessary to assess generalizability and robustness. This study should be considered as an early-stage feasibility demonstration rather than a deployment-ready system. Future work will extend the evaluation framework to include operational performance metrics such as travel time, energy efficiency, and safety margins, especially when migrating to physics-based simulators or real-world datasets.

6. Declarations

6.1. Author Contributions

Conceptualization: A.P.W., S., and A.W.; Methodology: S.; Software: A.P.W.; Validation: A.P.W., S., and A.W.; Formal Analysis: A.P.W., S., and A.W.; Investigation: A.P.W.; Resources: S.; Data Curation: S.; Writing Original Draft Preparation: A.P.W., S., and A.W.; Writing Review and Editing: S., A.P.W., and A.W.; Visualization: A.P.W. All authors have read and agreed to the published version of the manuscript.

6.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

6.3. Funding

This work was funded by the Ministry of Higher Education, Science, and Technology, Directorate General of Research and Development, through the Regular Fundamental Research Program for the 2025 Fiscal Year.

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M. Elassy, M. Al-Hattab, M. Takruri, and S. Badawi, "Intelligent transportation systems for sustainable smart cities," *Transportation Engineering*, vol. 16, no. 6, pp. 1-22, 2024.

[2] L. Gherardini and G. Cabri, "Tn2he Impact of Autonomous Vehicles on Safety, Economy, Society, and Environment," *World Electric Vehicle Journal*, vol. 15, no. 12, pp. 1–18, 2024, doi: 10.3390/wevj15120579.

[3] O. Tengilimoglu, O. Carsten, and Z. Wadud, "Implications of automated vehicles for physical road environment: A comprehensive review," *Transportation Research Part E: Logistics and Transportation Review*, vol. 169, no. 1, pp. 1-19, 2023.

[4] M. Hamidaoui, "Survey of Autonomous Vehicles' Collision Avoidance Algorithms," *Sensors*, vol. 25, no. 2, pp. 1–34, 2025, doi: 10.3390/s25020395.

[5] D. Parekh, "A Review on Autonomous Vehicles: Progress, Methods and Challenges," *Electronics (Switzerland)*, vol. 11, no. 14, pp. 1–18, 2022, doi: 10.3390/electronics11142162.

[6] M. Reda, A. Onsy, A. Y. Haikal, and A. Ghanbari, "Path planning algorithms in the autonomous driving system: A comprehensive review," *Robotics and Autonomous Systems*, vol. 174, no. 4, pp. 104630, 2024

[7] F. Sana, N. L. Azad, and K. Raahemifar, "Autonomous Vehicle Decision-Making and Control in Complex and Unconventional Scenarios—A Review," *Machines*, vol. 11, no. 7, pp. 1–29, 2023, doi: 10.3390/machines11070676.

[8] Ł. Łach and D. Svyetlichnyy, "Comprehensive Review of Traffic Modeling: Towards Autonomous Vehicles," *Applied Sciences (Switzerland)*, vol. 14, no. 18, pp. 8-16, 2024, doi: 10.3390/app14188456.

[9] Z. K. Ali and A. M. Abdulazeez, "A Review on Deep Reinforcement Learning for Autonomous Driving," *Indonesian Journal of Computer Science*, vol. 13, no. 3, pp. 4018–4035, 2024

[10] R. Audinys, Ž. Šlikas, and J. Radkeviˇ, "Deep Reinforcement Learning for a Self-Driving Vehicle Operating Solely on Visual Information," *Electronic Journal of e-Learning*, vol. 14, no. 825, pp. 1–30, 2025.

[11] J. Wu, "Extracting apple tree crown information from remote imagery using deep learning," *Computers and Electronics in Agriculture*, vol. 174, no. 7, pp. 1-14, 2020, doi: 10.1016/j.compag.2020.105504.

[12] A. Khlifi, M. Othmani, and M. Kherallah, “A Novel Approach to Autonomous Driving Using Double Deep Q-Network-Bsed Deep Reinforcement Learning,” *Word Electric Vehicle Journal*, vol. 16, no. 138, pp. 1–17, 2025.

[13] J. Hu, “A survey of decision-making and planning methods for self-driving vehicles,” *Frontiers in Neurorobotics*, vol. 19, no. 2, pp. 1-24, 2025, 2025, doi: 10.3389/fnbot.2025.1451923.

[14] Z. Lu, I. Afridi, H. J. Kang, I. Ruchkin, and X. Zheng, “Surveying neuro-symbolic approaches for reliable artificial intelligence of things,” *Journal of Reliable Intelligent Environments*, vol. 10, no. 3, pp. 257–279, 2024, doi: 10.1007/s40860-024-00231-1.

[15] D. Yan, Q. Guan, B. Ou, B. Yan, Z. Zhu, and H. Cao, “A Deep Reinforcement Learning-Based Decision-Making Approach for Routing Problems,” *Applied Sciences*, vol. 15, no. 9, pp. 1-12, 2025, doi: 10.3390/app15094951.

[16] R. Bin Issa, “Double deep Q-learning and faster R-CNN-based autonomous vehicle navigation and obstacle avoidance in dynamic environment,” *Sensors*, vol. 21, no. 4, pp. 1–24, 2021, doi: 10.3390/s21041468.

[17] F. WANG, X. ZHU, Z. ZHOU, and Y. TANG, “Deep-reinforcement-learning-based UAV autonomous navigation and collision avoidance in unknown environments,” *Chinese Journal of Aeronautics*, vol. 37, no. 3, pp. 237–257, 2024.

[18] Z. Zhang, H. Fu, J. Yang, and Y. Lin, “Deep reinforcement learning for path planning of autonomous mobile robots in complicated environments,” *Complex & Intelligent Systems*, vol. 11, no. 6, pp. 277, 2025, doi: 10.1007/s40747-025-01906-9.

[19] L. Sid Ahmed Abdellahi, Z. Zoubeir, Y. Mohamed, A. Haouba, and S. Hmetty, “Deep Reinforcement Learning for Optimal Replenishment in Stochastic Assembly Systems,” *Mathematics*, vol. 13, no. 14, pp. 1-12, 2025, doi: 10.3390/math13142229.

[20] M. Patil, P. Tambolkar, and S. Midlam-Mohler, “Optimizing Traffic Routes With Enhanced Double Q-Learning,” *IET Intelligent Transport Systems*, vol. 19, no. 1, pp. 1-12, 2025

[21] B. Petryshyn, S. Postupaiev, S. Ben Bari, and A. Ostreika, “Deep Reinforcement Learning for Autonomous Driving in Amazon Web Services DeepRacer,” *Information (Switzerland)*, vol. 15, no. 2, pp. 113-124, 2024, doi: 10.3390/info15020113.

[22] Y. T. Quek, L. L. Koh, N. T. Koh, W. A. Tso, and W. L. Woo, “Deep Q-network implementation for simulated autonomous vehicle control,” *IET Intelligent Transport Systems*, vol. 15, no. 7, pp. 875–885, 2021.

[23] X. Liu, “Toward the unification of generative and discriminative visual foundation model: a survey,” *Visual Computer*, vol. 41, no. 5, pp. 3371 – 3412, 2025, doi: 10.1007/s00371-024-03608-8.

[24] H. Hassani, S. Nikan, and A. Shami, “Improved exploration–exploitation trade-off through adaptive prioritized experience replay,” *Neurocomputing*, vol. 614, no. 1, pp. 128836, 2025.

[25] Í. Elguea-Aguinaco, A. Serrano-Muñoz, D. Chrysostomou, I. Inziarte-Hidalgo, S. Bøgh, and N. Arana-Arexolaleiba, “A review on reinforcement learning for contact-rich robotic manipulation tasks,” *Robotics and Computer-Integrated Manufacturing*, vol. 81, no. 6, pp. 1-17, 2023.

[26] J. Chen, “Multi-Agent Deep Reinforcement Learning Cooperative Control Model for Autonomous Vehicle Merging into Platoon in Highway,” *World Electric Vehicle Journal*, vol. 16, no. 4, pp. 225-237, 2025, doi: 10.3390/wevj16040225.