applied sciences

MDPI

Article

Reinforcement Learning Decision-Making for Autonomous Vehicles Based on Semantic Segmentation

Jianping Gao, Ningbo Liu, Haotian Li, Zhe Li *, Chengwei Xie and Yangyang Gou

College of Vehicle and Traffic Engineering, Henan University of Science and Technology, Luoyang 471000, China
* Correspondence: 9906397@haust.edu.cn

Abstract: In the complex and stochastic traffic flow, ensuring safe driving requires improvements in perception and decision-making. This paper proposed a decision-control method that leveraged the scene perception and understanding capabilities of semantic segmentation networks and the stable convergence strategies of Deep Reinforcement Learning (DRL) algorithms to achieve more accurate and effective autonomous driving decision-control. Perception features obtained from cameras and sensors equipped with a semantic segmentation model were used as input for the intelligent agent. DRL algorithms were employed to update decisions based on reward feedback. Experimental results on the CARLA simulation platform demonstrated that the semantic segmentation network effectively identified obstacles, vehicles, and drivable areas, providing high-quality perception data input for the intelligent agent's decision-making model. Compared to the original algorithms, the proposed Double Deep Q-Network-Semantic Segmentation (DDQN-SS) and Proximal Policy Optimization-Semantic Segmentation (PPO-SS) increased the reward value by approximately 25% and enhanced driving stability by 14.2% and 28.5%, respectively, enabling more stable and precise decision-control during driving. The method proposed in this paper has better improved the decision-control performance of PPO and DDQN in complex scenarios.

Keywords: autonomous driving; semantic segmentation; deep reinforcement learning; proximal policy optimization; double deep Q-network; CARLA simulation simulator

Academic Editor: Douglas O'Shaughnessy

Received: 27 December 2024
Revised: 20 January 2025
Accepted: 23 January 2025
Published: 27 January 2025

Citation: Gao, J.; Liu, N.; Li, H.; Li, Z.; Xie, C.; Gou, Y. Reinforcement Learning Decision-Making for Autonomous Vehicles Based on Semantic Segmentation. *Appl. Sci.* **2025**, *15*, 1323. https://doi.org/10.3390/app15031323

1. Introduction

In the development of autonomous driving technology, safety is the primary consideration. Decision-control, as one of the core modules of autonomous driving, serves as a bridge between perception and execution, directly influencing the vehicle's behavior decisions and safety in complex traffic environments. To prove that autonomous vehicles were safer than human drivers, approximately five billion miles of road testing were required [1], which consumed substantial human, material, and time resources for real-world testing. With the continuous development of autonomous driving technology, simulation experiments not only save resources but also allow for the simulation of various complex and dangerous traffic scenarios in a virtual environment [2]. Powerful perception and understanding capabilities provide accurate environmental information for simulation-based decision-control, thereby supporting complex decision-making processes. In this process, the application of semantic segmentation network technology becomes an important means of achieving environmental understanding.

Semantic segmentation, as one of the technologies in intelligent driving perception and advanced driver-assistance systems (ADAS) [3], has played a role in classifying environmental elements in scene images into various category labels [4]. Compared to single-object

recognition tasks and image segmentation, scene semantic segmentation technology has provided granular and high-level semantic information for subsequent scene analysis and visual understanding. This has helped autonomous vehicles simplify the extraction of environmental features and achieve an understanding of complex scenes during driving decision-making [5]. In the application of semantic segmentation in autonomous driving scenarios, it was necessary to design semantic segmentation models that balance efficiency and accuracy. Y. Zhu et al. [6] proposed a video prediction-based method that expanded the training set by synthesizing new training samples to improve the accuracy of semantic segmentation networks. This method utilized the ability of video prediction models to predict future frames to generate future labels. C. Liu et al. [7], addressing the issue of semantic segmentation networks being unable to effectively balance segmentation efficiency and accuracy, designed a lightweight semantic segmentation network, the LMBA convolutional unit (LCV), which enabled efficient feature extraction and helped semantic segmentation networks achieve better perception and recognition capabilities in autonomous driving scenarios. However, although the aforementioned studies have improved the accuracy of semantic segmentation in specific scenarios, they have not thoroughly explored its application in perception and recognition within complex road environments, nor have they comprehensively evaluated its practicality and stability. Common perception modules typically relied on raw images or object detection using bounding boxes, which introduced significant limitations in terms of efficiency and semantic richness. Raw images have high dimensionality, making feature extraction computationally expensive and slow to converge. Bounding box methods, while identifying the position and category of objects, failed to capture the detailed spatial and semantic information needed for complex decision-making tasks in autonomous driving. Semantic segmentation addressed these gaps by providing pixel-level classification of the entire scene. This allowed for precise identification of object contours and enabled the separation of drivable and non-drivable areas, pedestrians, vehicles, and other key scene elements. Compared to bounding box methods, semantic segmentation provided richer global scene understanding, reduced input complexity, and enhanced decision robustness, especially in dynamic and complex environments.

Traditional autonomous driving systems were rule-based. Due to the complexity of scene structural information, constructing rules was challenging, making it difficult to form complete and effective system decision models [8]. Therefore, vehicles in the past primarily relied on advanced driver-assistance systems (ADAS) to actively improve vehicle safety by assisting in acceleration/deceleration and steering control to reduce accidents and injuries [9]. For example, Zhu et al. [10] proposed a car-following model that worked in a human-like manner, utilizing speed, relative speed, and vehicle distance. However, ADAS could only be applied to relatively simple scenarios, such as highways. With the development of artificial intelligence in the field of autonomous driving, the understanding and decision-making in complex scenarios were synthesized by neural networks, and traditional advanced driver-assistance systems transitioned to deep reinforcement learning (DRL) agent-based decision-control methods. Reinforcement learning decision algorithms had the advantages of a simple structure and excellent performance [11]. Some studies combined deep reinforcement learning methods with autonomous driving to achieve advanced decision-making in simulation tests [12]. Agarwal T. et al. [13] proposed a framework that combined modular approaches with DRL methods to generate DRL strategies for urban driving tasks such as vehicle following and intersection navigation. Tao Xu et al. [14] proposed an end-to-end autonomous driving decision-making method based on an improved TD3 algorithm. This method used forward-facing cameras to capture data and introduced a new critic network to form a triple-critic structure, combining it with target-maximization operations to address the underestimation problem in the TD3 algorithm.

Kendall A. et al. [15] utilized forward-facing camera images, current vehicle speed, and steering angles as inputs to train lane-keeping strategies, demonstrating the applicability of DRL in real-world autonomous driving scenarios. MP Ronecker et al. [16] introduced a method for safely navigating autonomous vehicles on highways by combining insights from deep Q-networks and control theory. The deep Q-network was trained in simulations and acted as the central decision-making unit by setting targets for the trajectory planner.

However, traditional rule-based autonomous driving systems have not been able to effectively construct decision models for complex road conditions, making it difficult to address decision-making demands in urban traffic scenarios. Other reinforcement learning studies have mostly focused on the comparative evaluation of DRL algorithm strategies, which has impacted decision-making accuracy. Given this, this paper integrates semantic segmentation technology with a DRL decision model to propose a decision-control method. It fully leverages semantic segmentation to enhance perception performance, thereby helping the DRL algorithm capitalize on its advantages in strategy update stability and balancing short-term and long-term rewards, ultimately strengthening autonomous driving decision-making capabilities in complex urban traffic environments. This study generates high-quality perception data through semantic segmentation, which provides pixel-level classification of the environment, offering detailed semantic information about drivable areas, obstacles, and dynamic objects. Rule-based methods, due to their reliance on predefined logic, lack of semantic understanding, and inflexibility, are inherently limited in handling complex and dynamic environments. These methods often fail to generalize to new scenarios and are sensitive to noise and edge cases. Semantic segmentation allows for a more comprehensive understanding of the scene, enabling the RL agent to effectively perceive and respond to dynamic and unpredictable situations.

This study generates high-quality perception data through semantic segmentation, simplifying the complexity of the state space and significantly improving the efficiency and performance of the reinforcement learning algorithms. By enhancing environmental perception capabilities with the semantic segmentation network and tightly integrating it with the decision-making control of the reinforcement learning algorithm, a self-driving decision-control framework is formed. This framework not only improves perception accuracy but also reduces the interference of low-quality perception data in the decision-making process, significantly enhancing the system's robustness and adaptability in complex urban driving environments.

2. Materials and Methods

2.1. Semantic Segmentation Network

Semantic segmentation classified each pixel in an image to separate and label scene elements such as roads, traffic signs, pedestrians, and obstacles. This provided critical semantic information for the decision-making system to identify drivable areas and potential hazards. With the development of Convolutional Neural Networks (CNN), semantic segmentation models based on fully convolutional networks achieved high segmentation performance. The semantic segmentation model used in this paper was DeepLabv3P [17].

Compared to other FCN-based methods, DeepLabv3P uses Atrous convolution and the ASPP module, providing better multi-scale feature extraction capabilities. This is crucial for capturing small objects (such as pedestrians) and large background structures (such as road boundaries), making it highly suitable for the diversity and dynamic environments of autonomous driving. DeepLabv3P includes an additional decoder that enhances edge detail preservation and improves the segmentation of small objects and fine boundaries. This is especially beneficial for tasks such as recognizing drivable areas and obstacles. DeepLabv3P demonstrates better generalization capabilities in handling complex scenes, making it more

reliable in dynamic traffic environments. The DeepLabv3P network processes input images through stacked convolutional layers and atrous convolutions, ultimately producing a pixel-level prediction map of the same size as the input image. Its overall structure is shown in Figure 1.

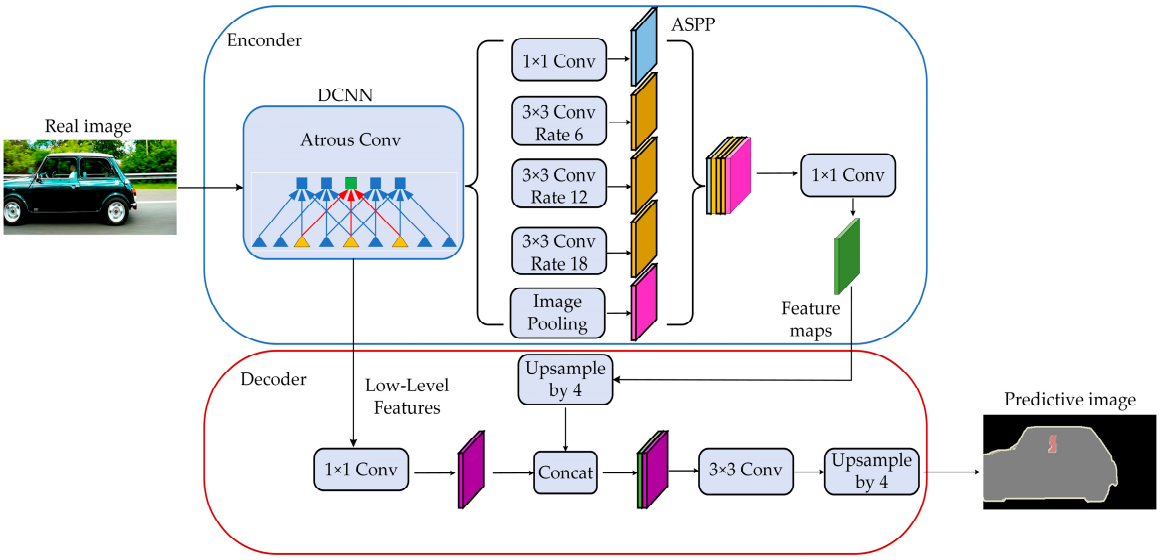


Figure 1. DeepLabv3P Network Architecture.

The Encoder consisted of a backbone network (DCNN) with atrous convolutions, and its backbone structure used an improved Xception model. Following it was the Atrous Spatial Pyramid Pooling (ASPP) module, which introduced multi-scale information. The network structure also included a Decoder module that further fused feature maps to optimize segmentation performance, particularly at the edges of targets. Additionally, the DeepLabv3P network applied Depthwise Separable Convolution to the ASPP and Decoder modules, improving the stability and efficiency of semantic segmentation. The Encoder provided two inputs to the Decoder: the first was feature maps obtained through the ASPP module and processed by an additional convolutional layer; the second was low-level features.

The Decoder received two outputs from the Encoder. It applied a 1×1 convolution to the input low-level features for channel reduction. The feature maps from the ASPP were resized through interpolation to match the dimensions of the low-level features and then concatenated with the channel-reduced low-level features. The combined features were then processed by a set of 3×3 convolutional layers. Finally, linear interpolation upsampling was used to generate a prediction label map with the same size as the original image.

The input images for training the semantic segmentation model were 512×512 pixels. The model was trained using an SGD optimizer with an initial learning rate of 0.01 and a polynomial decay scheduler with a batch size of 4. The training was carried out for 70,000 iterations. The inference speed of the DeepLabv3P model on 512×512 input was 50 ms per frame.

2.2. Reinforcement Learning (RL)

With the increasing complexity and dynamic changes in traffic environments, reinforcement learning (RL) gradually became a widely recognized technology to enhance the adaptability and decision-making capabilities of autonomous driving systems. By simulating feedback mechanisms in real driving environments, reinforcement learning enabled

autonomous driving systems to independently learn and optimize decision-making strategies in ever-changing traffic conditions, thereby improving their intelligence and autonomy.

Reinforcement learning improves an agent's behavior in completing tasks through continuous interactions with the environment, similar to the learning patterns of humans or biological entities. The relationship between the agent's decision model and the test environment was modeled as a Markov Decision Process (MDP) [18]. MDP was defined as a five-tuple $\langle S, A, P, R, \gamma \rangle$, where S represented a finite set of states, A represented the set of actions, $P(s'|s,a)$ denoted the state transition probability distribution, $R(s,a)$ denoted the reward function, and $\gamma \in [0,1]$ was the discount factor used to balance short-term and long-term rewards. At time t, the agent perceived the current state s_t of the environment, selected an action a_t based on its policy, and received an immediate reward r_t and a new state s_{t+1} from the environment. The goal of reinforcement learning was to optimize the policy to maximize the cumulative discounted reward for the agent.

2.2.1. Double Deep Q Networks (DDQN)

DDQN could effectively cope with the uncertainty and complexity in autonomous driving environments, enabling autonomous vehicles to make appropriate decisions when facing constantly changing road and traffic conditions. DDQN was an improved algorithm based on Deep Q-Network (DQN) [19], designed primarily to solve the problem of overestimation of target Q-values in DQN. The core idea of DDQN was to separate action selection from target Q-value estimation by introducing a prediction network and a target network to reduce estimation bias. First, the corresponding action $a_{\max}$ was selected through the prediction network $Q(s,a;\theta)$. The formula is expressed as follows:

$$a_{\max} = \text{argmax}_a Q(s_{t+1}, a; \theta) \tag{1}$$

In the formula, s_{t+1} is the state at the time $t+1$, a is the action taken, and θ is the online weight parameter of the network used in action selection.

The calculation of the target Q-value relied on the collaboration between the target network and the prediction network, and its formula is as follows:

$$y_i = r_{t+1} + \gamma Q\left[s_{t+1}, \text{argmax}_a Q(s_{t+1}, a; \theta_t); \theta_t^-\right] \tag{2}$$

In the formula, r_{t+1} is the immediate reward obtained by the agent from the environment after taking action a_t at time t, γ is the discount factor that controls the influence of future rewards on current decisions, and $\gamma \in [0,1]$; $Q[s_{t+1}, a; \theta_t]$ is the Q-value estimate of taking action a at time t in state s_{t+1}, predicted by the prediction network. θ_t is the parameter of the prediction network at time t, and $\text{argmax}_a Q(s_{t+1}, a; \theta_t)$ represents the action with the maximum Q-value among all actions a in state s_{t+1}, calculated based on the parameters θ_t of the prediction network at time t. The parameters θ_t^- of the target network are periodically updated by copying the parameters θ_t of the prediction network; thus, $\theta_t^- = \theta_t$.

The agent interacted with the environment and generated and stored tuples $(s_t, a_t, r_{t+1}, s_{t+1})$ in the experience replay buffer. The prediction network selected actions using the ε-greedy strategy. After the prediction network determined the actions, the target network collaborated to calculate the target Q-values. The prediction network's parameters were updated through the loss function, and the parameters of the prediction network were synchronized to the target network every fixed number of steps. This process was repeated iteratively to optimize the agent's decision-making ability while reducing the overestimation of target Q-values by separating the prediction network and the target network. The framework of the DDQN algorithm is shown in Figure 2.

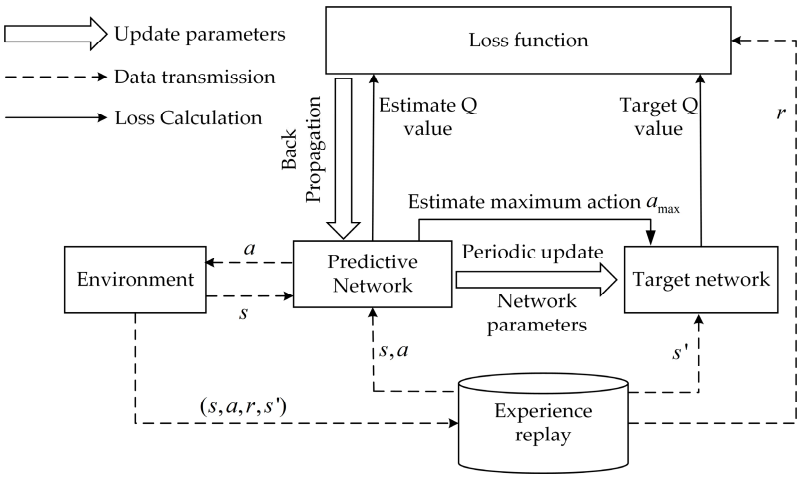


Figure 2. Double Deep Q-Network algorithm block diagram.

2.2.2. Proximal Policy Optimization Algorithm (PPO)

In autonomous driving, PPO handled continuous control actions better than DDQN. PPO supported autonomous driving systems in learning decisions more effectively through a stable policy optimization method, thereby improving the safety and robustness of the systems. PPO was a policy gradient-based algorithm that directly optimized the policy function. The policy function $\pi(a|s)$ represented the probability of taking action a in state s. In autonomous driving scenarios, PPO learned the optimal probability distribution of selecting different driving actions given the vehicle's state. The PPO algorithm used policy gradient-based optimization to maximize the expected cumulative return of the policy function. Its objective function based on the policy gradient method is:

$$L^{PG}(\theta) = \hat{E}_t\left[\log \pi_\theta(a_t|s_t)\hat{A}_t\right] \tag{3}$$

In the formula, θ represents the parameters of the current policy, $\hat{E}_t$ denotes the expectation operation at time step t, and $\log \pi_\theta(a_t|s_t)$ is the logarithm of the policy output, which is the probability of an action based on the observed state as input. a_t and s_t represent the action and state, respectively, and $\hat{A}_t$ denotes the advantage function at time step t, which measures the advantage of selecting action a_t in state s_t compared to other actions.

To prevent the policy updates from being overly large and causing significant deviations in vehicle actions or trajectories, the environment for simulating the PPO algorithm was introduced into the simulation platform. The PPO algorithm incorporated a clipped objective function $L^{CPI}(\theta)$ to limit the extent of policy updates. The formula is:

$$L^{CLIP}(\theta) = E_t\left[\min\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{4}$$

In the formula, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi\theta_{\text{old}}(a_t|s_t)}$ is the probability ratio of the new policy to the old policy, which measures the extent of policy changes. ε is a hyperparameter that represents the maximum extent to which the policy ratio is allowed to deviate from 1. $clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)$ is the clipping function that restricts $r_t(\theta)$ to the interval $[1-\varepsilon, 1+\varepsilon]$.

PPO combines policy optimization and value function optimization, and the final objective function is:

$$L^{PPO}(\theta) = E_t\left[L^{CLIP}(\theta) - c_1 \cdot E_t\left(V_\theta(s_t) - V_t^{target}\right)^2 + c_2 \cdot H(\pi_\theta)\right] \tag{5}$$

In the formula, $L^{CLIP}(\theta)$ is the clipped objective function, which optimizes the cumulative reward of the policy. $\left(V_\theta(s_t) - V_t^{target}\right)^2$ is the loss of the value function, minimizing the prediction error of the state value. $H(\pi_\theta)$ is the entropy regularization term of the policy, which increases the exploration capability of the policy. c_1, c_2 are the weight coefficients, controlling the influence of each part.

The algorithm framework of PPO is shown in Figure 3. The Actor used the old policy and the new policy to obtain the probability ratio of a specific action for each policy, and the ratio was used in the clipped objective function. The Critic predicted the state value $V(s_t)$ for a given state s_t, which was used to evaluate the agent's expected long-term returns in that state and calculate the advantage function $\hat{A}_t$ for the clipped objective function. Finally, the clipped objective function provided new policy updates.

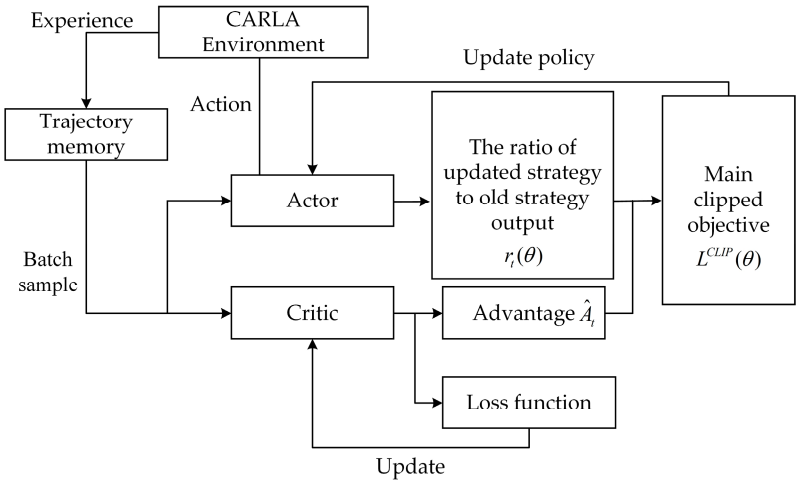


Figure 3. Proximal Policy Optimization algorithm block diagram.

2.3. Agent Model

The simulation platform supported API interface connections, allowing the integration of the agent model with the platform for training. For the DDQN and PPO algorithms, the agent could be trained simply by replacing the source code of the agent model with one algorithm or the other. The agent model primarily consisted of several key components: state space, action space, and reward mechanism.

2.3.1. State Space

The state space input included a single-channel semantic label map generated by the semantic segmentation model, with pixel categories covering road boundaries, lane lines, drivable and non-drivable areas, as well as on-road elements (traffic participants, obstacles), and off-road elements (buildings, trees, etc.). The single-channel semantic label maps generated by the DeepLabv3P network were downsampled to a fixed resolution (256×256 pixels) and then fed into the CNN used by the reinforcement learning agent. This method ensured computational efficiency, reduced memory consumption, and retained the essential spatial and semantic information required for decision-making. Additionally, it included information such as vehicle speed and distance provided by radar and the Global Navigation Satellite System (GNSS). GNSS and radar data were fused as additional numerical features with the features extracted by the CNN from the semantic label map. Specifically, the CNN processed the single-channel semantic label map to extract spatial and semantic features. At the same time, GNSS and radar data, including vehicle speed, distance to obstacles, and lateral deviation, were represented as normalized numerical vectors. During the fully connected layer stage, these numerical features were concatenated

with the CNN features, forming the complete state space for the reinforcement learning agent. The state set is represented as:

$$s = (v_{ego}, L_{ego}, D_{ego}, d_c, v_{adv}, d_{adv}, M) \quad (6)$$

In the formula, v_{ego} is the speed of the ego vehicle, L_{ego} is the position of the ego vehicle, and D_{ego} is the distance between the ego vehicle and the lane line. d_c is the lateral distance between the vehicle and the lane centerline, v_{adv} is the speed of other vehicles, d_{adv} is the distance to other vehicles, and M is the single-channel label map generated by semantic segmentation.

2.3.2. Action Space

In the agent model, the action space consisted of two actions: speed control and steering. In simulation tests, the low-level controller executed the decision actions to ensure smooth driving of the test vehicle. These actions included interactions involving the ego vehicle in straight driving, turning conditions, and interactions with other vehicles, which were learned and determined by the deep reinforcement learning algorithm based on the input from the state space. The action space was defined as: $A = (a_1, a_2)$, where a_1 was speed control, ranging from 0 m/s to 12.5 m/s; a_2 was steering control, with a turning angle set between 30° and 60°.

2.3.3. Incentive Mechanism

The reward mechanism determined the convergence speed and decision accuracy of the agent model. The following were the five designed reward mechanism indicators:

(1) Safe distance threshold R_o and collision reward R_c

Reward outcome for evaluating the relationship between the ego vehicle and obstacles R_o:

$$R_o = \begin{cases} C \cdot \omega, \min(d_{adv1}, d_{adv2}) > T \\ -C \cdot \omega, \text{otherwise} \end{cases} \quad (7)$$

In the formula, C is a constant representing the baseline magnitude affecting this component of the total reward, ω is the adjustment weight of the constant C, and T is the threshold for determining whether the vehicle maintains a safe distance from obstacles ahead [20]. $\min(d_{adv1}, d_{adv2})$ is the minimum distance between the ego vehicle and the detected obstacle.

The reward value R_c was assigned a negative value when the autonomous vehicle collided.

Its calculation formula is as follows:

$$R_c = \begin{cases} -C_c \cdot \omega_c, \text{Collision with an obstacle} \\ 0, \text{otherwise} \end{cases} \quad (8)$$

In the formula, C_c is a constant representing the baseline magnitude affecting the total reward; ω_c is the adjustment weight of C_c.

(2) Lane deviation reward R_i

The correctness of the ego vehicle's driving on the lane was determined by the distance difference, calculating the lateral distance (perpendicular to the driving direction) between the vehicle's current position and the lane centerline. The smaller the distance, the higher

the reward given, with the reward reaching its maximum value when the distance was 0 (exactly on the centerline). Its calculation formula is:

$$R_i = \max\_R - k \times |d_c| \tag{9}$$

In the formula, max_R is the maximum reward value set when the vehicle is exactly on the centerline, k is a scaling function used to adjust the sensitivity of the reward to the distance, and d_c is the lateral distance between the vehicle's current position and the lane centerline.

(3) Speed and steering angle variation rewards R_s, R_t

The driving stability of the self-driving vehicle in the test scenario needs to be given to the driving direction and driving speed of the two indicators to set the appropriate reward values R_s, R_t. As such, the model in the training process controls the driving direction and speed of the self-driving vehicle without excessive deviation, ensuring the stability of the driving vehicle. If the direction and speed of the two changes, compared to the previous, did not occur with many deviation changes, the reward value feedback to the model is high, and if vice versa, the value feedback is low. Therefore, the formula for the two reward values is:

$$R_s = (T_s - |Sr_t - Sr_{t-1}|) \times w_s \tag{10}$$

$$R_t = (T_t - |Th_t - Th_{t-1}|) \times w_t \tag{11}$$

In the formula, T_s represents the threshold for whether the vehicle suddenly makes a sharp change in direction. Sr_t and Sr_{t-1} represent the steering angles of the vehicle at time t and $t-1$ ($t > 0$), respectively. w_s is the adjustment weight for the steering reward value. T_t is the threshold for determining whether the vehicle suddenly makes a sharp change in speed. Th_t and Th_{t-1} represent the vehicle's speed at time t and $t-1$ ($t > 0$), respectively, and w_t is the adjustment weight for the vehicle's speed reward value.

The total reward value was composed as the sum of the above five reward values. The equation is:

$$R_A = R_o + R_i + R_c + R_s + R_t \tag{12}$$

3. Experimental Procedure and Results

3.1. Decision-Control Framework for Autonomous Vehicles

To achieve accurate and stable decision-control for autonomous vehicles, this paper proposed a decision-control framework, as shown in Figure 4. First, key elements and road outlines were extracted from real road scenes to generate road maps, which were then imported into the simulation platform as scene models. The semantic segmentation neural network was trained using real road image datasets, achieving good segmentation results. The pre-trained semantic segmentation model loaded its weights through CARLA's Python API and connected with the simulation platform. Cameras were installed on the ego vehicle, and the RGB images captured by the cameras were identified and predicted by the pre-trained semantic segmentation model, extracting different types of scene elements from the images and completing label classification. Road visual perception data were simplified to extract drivable areas and reduce environmental complexity. The output of the semantic segmentation model (single-channel semantic label map) was downsampled and then fed into the neural network of the reinforcement learning (RL) agent for state embedding. Subsequently, measurement data were obtained via radar and the Global Navigation Satellite System (GNSS) in CARLA, including vehicle speed, distances to other vehicles, lane markings, collision information, and obstacle positions. These data constituted the state space input for the agent model. In the proposed framework, speed and collision data

were combined as part of the state representation and fused with the features extracted from the semantic label map. The fusion process was as follows: a convolutional neural network (CNN) was used to extract semantic segmentation features from the single-channel semantic label map. The CNN output was a flattened feature vector representing the spatial and semantic context of the environment. Numerical features, such as speed, collision data, and distance to obstacles, were preprocessed and normalized into low-dimensional vectors. In the fully connected (FC) layer of the reinforcement learning (RL) agent's neural network, the semantic feature vector and the numerical feature vector were concatenated, ensuring that both types of features were jointly considered in the decision-making process.

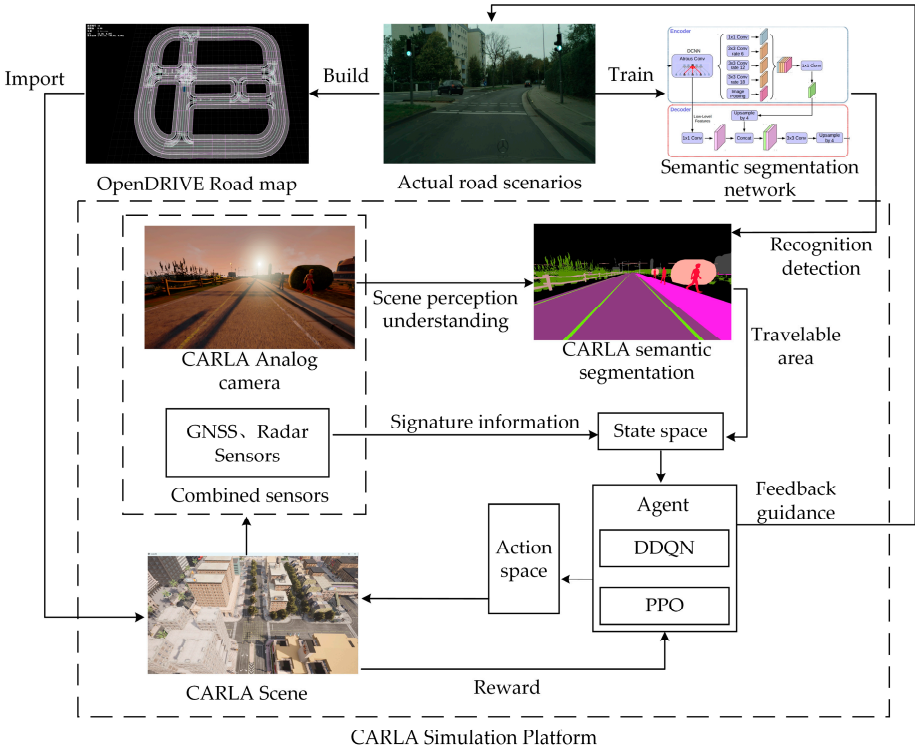


Figure 4. Decision-control architecture.

The agent received state space information and determined decisions in the action space, such as steering and speed control, which were then executed and presented in the CARLA environment. Reward values were generated through the reward mechanism and fed back to the agent to update future decisions. Finally, the decision feedback from the agent in the simulation environment guided the driving decisions of the autonomous vehicle on real roads. According to the framework design, the training was set as a closed-loop system, enabling the agent to learn from the environment and improve its driving strategy in the test scenarios.

3.2. CARLA Simulator

The simulation platform in this paper was the open-source CARLA autonomous driving simulator, which included a modular and flexible API designed to address a series of tasks related to autonomous driving. The CARLA simulator supported the configuration of various sensors, including cameras, radar sensors, and others. This study used CARLA version 0.9.15, a widely used open-source autonomous driving research simulation platform. The simulation environment was configured as shown in Table 1.

Table 1. Sensor parameters.

	Sensor Type	Parameters
Sensor	RGB camera	Resolution of 512 × 512 FOV (Field of View) of 90Frequency of 20 Hz
	Radar	50 Hz, 100 m range
	GNSS sensor	10 Hz

The CARLA simulator (version 0.9.15) was used to generate high-resolution RGB images and corresponding pixel-perfect semantic annotations under various environmental conditions, including different weather, lighting, and traffic scenarios. These synthetic data provided a scalable and cost-effective method for training segmentation networks, especially for rare and hazardous scenarios that are difficult to capture in real-world datasets. The real-world images came from publicly available datasets, which included complex real driving scenes with semantic annotations. These datasets complemented the synthetic CARLA data by introducing natural variations, noise, and edge cases common in real-world driving.

In our study, the CARLA environment was customized to address the inherent class imbalance in driving scenes, where large areas (such as roads) dominated the dataset, while smaller safety-critical objects (such as pedestrians and vehicles) were underrepresented. To solve this issue, this paper implemented domain-specific adjustments: during the training of the DeepLabv3P segmentation network, this paper applied class weighting in the cross-entropy loss function. Class weights were calculated based on the inverse frequency of each class in the CARLA dataset. This ensured that underrepresented classes, such as pedestrians and vehicles, received higher weights, encouraging the network to focus more on these classes. The simulation experiments in this paper were conducted under ideal conditions, primarily during clear noon conditions, to ensure that lighting and weather variations were minimized, reducing their impact on segmentation quality.

3.3. Scenario Generation for Simulation Platforms

To test driving decisions, specific scenarios were first constructed. Key elements and structural outlines were extracted from real road scenes. Maps were generated in RoadRunner, exported in OpenDRIVE format, and imported into CARLA to create a simulation environment. The ego vehicle drove in the simulation scenario, while other vehicles randomly chose paths and traveled at safe distances and restricted speeds. The complex scenes were created in the CARLA simulator (version 0.9.15), as shown in Figure 5 below.

These scenes were designed to replicate real-world driving challenges and evaluate the decision-making performance of the proposed framework. The generation process involved constructing realistic road networks (such as urban intersections and multi-lane roads), simulating dynamic traffic with varying densities, and including roads with T-junctions and four-way intersections, requiring the agent to handle tasks like merging, yielding, and prioritizing its path relative to other vehicles. Multi-lane roads with two to four lanes in each direction were also included, simulating both urban and highway driving conditions. Traffic density varied dynamically, with scenes ranging from low-density traffic (representing less congestion) to high-density traffic (simulating limited maneuvering space during urban peak hours). In addition, traffic lights at intersections were included to simulate complex urban traffic scenarios. Pedestrians were included in some scenes, crossing the road at designated crosswalks, adding an extra challenge in avoiding collisions. To ensure comprehensive scene coverage, the scenes were classified based on road complexity, traffic density, and environmental conditions. About 30% of

the scenes involved low-density traffic, 50% involved medium-density traffic, and 20% involved high-density traffic. At least 40% of the scenes included other challenges, such as intersections, traffic lights, crosswalks, and obstacles.

Figure 5. Scenes with random traffic flow.

3.4. Simulation Parameter Setting

The semantic segmentation algorithm was trained with 70,000 iterations, a batch size of 4, a minimum scaling factor of 0.5, a maximum scaling factor of 2.0, and a scaling step size of 0.25. The input of the agent training model was set to driving perception data and features such as speed, while the output was set to actions such as speed control and steering.

The learning rates for both the PPO and DDQN algorithms were set to 0.0001, with a training set length of 7500, a batch size of 64, and a discount factor of 0.99. The number of training episodes for both was 200,000. When the ego vehicle deviated from the lane or collided with an obstacle, it was reset to its initial state. The target network update frequency for DDQN was 5, meaning the target network synchronized the online network's weights every five update cycles. PPO used action standard deviation adjustment to control exploration: the initial exploration noise was set to 0.2, the decay rate to 0.05, and the minimum exploration noise to 0.05. The initial standard deviation (0.2) allowed significant exploration, encouraging the agent to sample various actions and gather comprehensive environmental feedback. The action standard deviation decay rate of 0.05 controlled the exploration noise during training, and this gradually decreasing exploration allowed the agent to focus on developing and learning the optimal strategy. When the standard deviation reached the pre-determined minimum value (0.05), the decay stopped to ensure the agent maintained a small degree of flexibility during decision-making. The detailed parameter settings are shown in Table 2.

Table 2. Simulation experiment parameter setting.

	Training Parameters	Numerical Value
Semantic segmentation parameter settings	Batch size	4
	Minimum scaling factor	0.5
	Maximum scaling factor	2.0
	Scaling step	0.25
	Number of iterations	70,000
RL algorithm parameter settings	Learning rate	0.0001
	Batch size	64
	Discount factor	0.99
	Training set length	7500
	Number of training episodes	200,000
DDQN	Target network update frequency	5
PPO	Exploration noise	0.2
	Decay rate	0.05
	Minimum exploration noise	0.05

During simulation training, a car or a bus was randomly generated every 5 s to join the traffic flow and drove at a speed of 4–6 m/s, following traffic rules according to the Intelligent Driver Model [21].

3.5. Simulation Results and Discussion

3.5.1. Semantic Segmentation Network Training Curve Results

The evaluation of semantic segmentation focused on the accuracy and mean intersection over union (MIOU) metrics [22]. Accuracy measured segmentation precision, defined as the ratio of correctly classified pixels to the total number of pixels; the higher this ratio, the better the segmentation performance. MIOU evaluated the segmentation accuracy and completeness for each category by calculating the ratio of the intersection to the union between predictions and ground truth labels. Together, these two metrics provided a quantitative basis for assessing the performance of semantic segmentation models. As seen in Figure 6a, accuracy showed an upward trend before 20,000 iterations, indicating that the model gradually improved its prediction accuracy for pixel categories. Although there were some fluctuations, the overall trend was upward, which indicated that the model's overall segmentation accuracy was improving. After 20,000 iterations, the accuracy curve stabilized, with values ranging between 0.8 and 1, demonstrating that the model had reached a stable classification capability on the current training data. Figure 6b shows that the MIOU curve gradually increased during training, indicating that the model's segmentation performance continuously improved. Combining the trends of accuracy and MIOU, it was evident that the model had achieved good training results.

In Figure 7, the left side showed the input test set images, while the right side displayed the generated prediction images. From the results, it could be observed that the semantic segmentation model effectively simplified, extracted, and categorized the predefined elements of semantic category scenes.

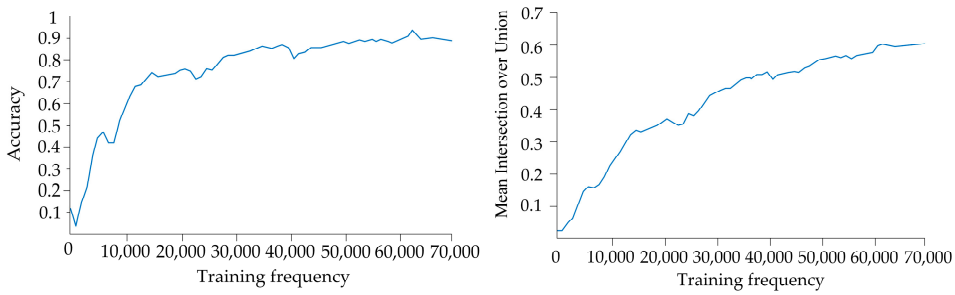


Figure 6. Semantic Segmentation Network Training Results. (**a**) The accuracy curve of the semantic segmentation network. (**b**) The average intersection and merger ratio curves for the semantic segmentation network.

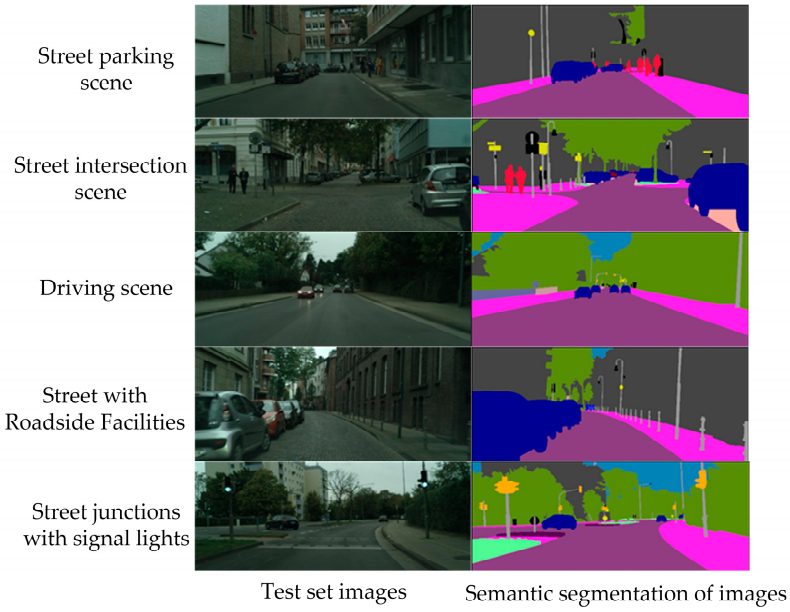


Figure 7. Comparison of test set images with predicted images using pre-trained semantic segmentation models.

Through the training process and results of the semantic segmentation model (DeepLabv3P), this paper identified several challenging categories and scenarios. These failure modes included pedestrians, bicycles, and distant vehicles, which were often misclassified or overlooked due to their small pixel representation in semantic label maps. Consistently segmenting moving vehicles and pedestrians proved difficult, especially in high-density traffic or when objects moved at high speeds. Overlapping or visually similar classes (e.g., curbs and sidewalks) occasionally caused boundary misclassification. Rain and low-light conditions reduced the visibility of lane markings and small obstacles, leading to segmentation errors. In traffic-dense urban intersections, the network sometimes struggled to segment traffic signs or partially occluded objects, such as pedestrians behind vehicles. These issues mainly stemmed from class imbalance, limited spatial resolution, and the domain gap between synthetic training data and real-world variability.

3.5.2. Agent Training Results

The agent, by utilizing a well-designed reward mechanism, continuously optimized its strategy during interactions with the environment and feedback processes, learning the optimal decisions in different states by maximizing rewards to achieve route objectives. The reward value was calculated based on the reward mechanism. The average deviation from

the center was used to measure the mean deviation relative to the lane center. The smaller this value, the more stable the driving, the more precise the actions, and the lower the path deviation, thereby providing a more comprehensive evaluation of the agent model's task performance.

The training focused on comparing the driving decision-control effects of two algorithms in a simulation environment, with and without the introduction of semantic segmentation perception and recognition functionality. By observing the reward values and the average deviation from the center in the agent model, the performance of the agent under different training methods was analyzed to determine the stability of autonomous driving decision-control. The decision effects of the DDQN-SS and PPO-SS methods introduced in this paper compared to the traditional DDQN and PPO are shown in the figure below. Figure 8a,b show that the DDQN-SS and PPO-SS methods with the semantic segmentation network outperformed the traditional DDQN and PPO algorithms in decision-control performance. In the 0–40,000 episodes, the reward values of PPO and DDQN fluctuated significantly, with multiple peaks and troughs. However, the reward values of DDQN-SS and PPO-SS exhibited smaller fluctuations compared to the original algorithms during this phase. As the training progressed to around 100,000 episodes, DDQN and PPO reached a higher reward value peak but then quickly dropped. Although DDQN-SS and PPO-SS did not reach the peak reward values of the original algorithms during this period, their decrease was smaller, and they remained relatively stable. In the 150,000–200,000 episodes, DDQN-SS and PPO-SS's reward values gradually surpassed the traditional algorithms, maintaining a relatively stable upward trend, while DDQN and PPO's reward values still experienced some fluctuations. The reward value fluctuations of DDQN-SS and PPO-SS were relatively small throughout the training process, especially in the later stages, demonstrating better stability.

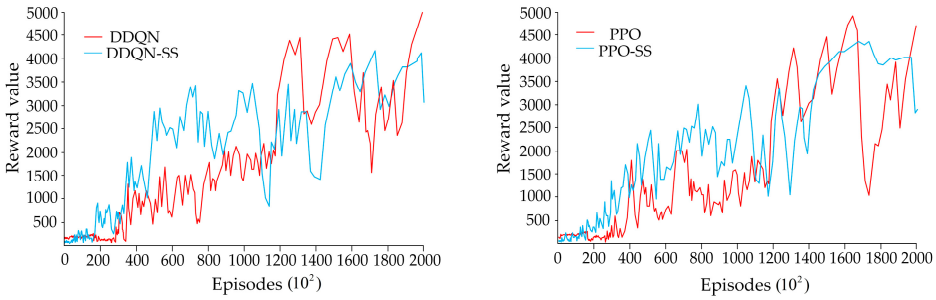
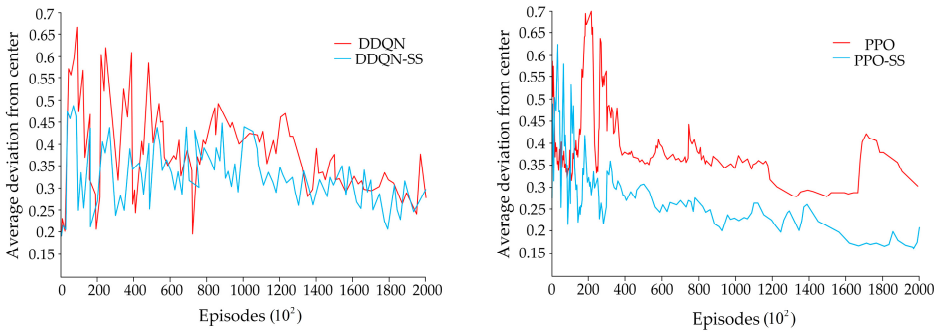


Figure 8. (**a**) Reward Values of DDQN and DDQN-SS. (**b**) Reward Values of PPO and PPO-SS.

In the early stages of training, the reward values of DDQN-SS and PPO-SS increased more rapidly, indicating that the semantic segmentation network significantly enhanced environmental perception capabilities and accelerated the learning efficiency of the algorithms. During the mid-training phase, the reward values of both methods remained at a high level, reflecting stronger adaptability to the environment. In the later stages of training, the fluctuation amplitude of the reward values significantly decreased, demonstrating higher decision stability and reliability.

Figure 9a,b show that the average deviation-from-center curves of PPO-SS and DDQN-SS showed an overall downward trend, indicating that the vehicles gradually maintained a better position near the center of the road as training progressed. In the later stages of training, the average deviation-from-center values of DDQN-SS and PPO-SS fluctuated less. Table 3 (Data Comparison of DDQN-SS, PPO-SS with DDQN, PPO) showed that their

variance values decreased, reflecting that the models gradually adapted and converged to better behavior, with a significant improvement in overall stability.



(**a**) (**b**)

Figure 9. (**a**) Average Deviation from Center of DDQN and DDQN-SS. (**b**) Average Deviation from Center of PPO and PPO-SS.

Table 3. Data Comparison of DDQN-SS, PPO-SS with DDQN, PPO.

	Episodes (10^2)	DDQN	DDQN-SS	PPO	PPO-SS
Reward Value Range	0–400	100–1200	100–1800	200–1800	100–1700
	400–800	500–1500	1200–3500	500–2100	1000–3000
	800–1600	1300–4500	1000–3700	700–4700	1000–4200
	1600–2000	1800–5000	3000–4000	1000–4500	2800–4500
Average Deviation from Center Range	0–400	0.2–0.66	0.25–0.48	0.3–0.7	0.22–0.63
	400–800	0.2–0.6	0.25–0.43	0.35–0.45	0.25–0.3
	800–1600	0.28–0.48	0.25–0.43	0.3–0.4	0.18–0.28
	1600–2000	0.25–0.38	0.2–0.35	0.3–0.43	0.15–0.25
Reward Value Variance	0–600	199,093.14	803,704.74	225,405.99	440,959.80
	600–1400	985,624.10	347,929.36	967,823.92	318,713.77
	1400–2000	587,908.79	392,691.37	839,591.20	203,899.19
Average Deviation from Center Variance	0–600	0.163	0.005	0.013	0.012
	600–1400	0.006	0.003	0.002	0.001
	1400–2000	0.001	0.001	0.003	0.001

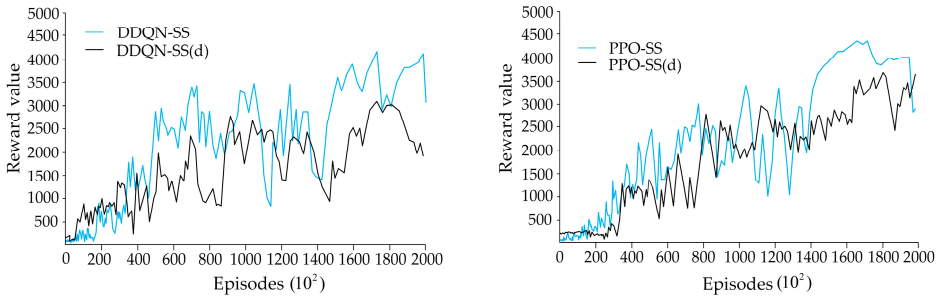
In Table 3, the variance values represented the degree of fluctuation in the reward values and average deviation from the center, which indicated the stability of the agent's decision-making process during training. Smaller variance values meant that the agent's decision-control was more stable and consistent, with less fluctuation during the training process; conversely, larger variance values indicated that the agent's decisions had greater uncertainty or volatility.

As shown in Table 3, in the range of 0–40,000 episodes, DDQN-SS and PPO-SS reached reward values between 100 and 1800, while DDQN and PPO reached reward values between 100 and 1200. This indicated that the integration of semantic segmentation improved the agent's ability to achieve maximum reward earlier in training. As the training progressed (40,000–80,000 and 80,000–160,000 episodes), the gap between the methods became more pronounced, with DDQN-SS and PPO-SS continuing to show significantly higher reward ranges, reinforcing the idea that semantic segmentation enhances the agent's

decision-making process by providing richer environmental information. In the range of 0–40,000 episodes, the deviation of DDQN-SS was between 0.25 and 0.48, while DDQN's deviation was between 0.2 and 0.66, reflecting better lane-keeping performance due to semantic segmentation. As training progressed, the deviation values of DDQN-SS and PPO-SS further decreased, indicating that, compared to DDQN and PPO, they performed better in lane centering, and their driving behavior was more stable, suggesting less fluctuation in their deviation from the center. This trend indicated improvements in decision-control, especially in maintaining stable trajectories in complex driving scenarios.

In the parts with smaller reward value fluctuations, DDQN-SS and PPO-SS stabilized at 2500, which was an increase of approximately 25% compared to the traditional algorithm's reward value of 2000. Regarding the average deviation-from-center values, DDQN-SS stabilized at 0.35 (a 14.2% improvement over DDQN), and PPO-SS stabilized at 0.25 (a 28.5% improvement over PPO), indicating that the proposed method demonstrated greater advantages in driving stability. A detailed data comparison between the relevant algorithms is shown in Table 3.

To assess the potential impact of segmentation accuracy on the overall performance of the proposed framework, an additional test was conducted to simulate the effect of different levels of semantic segmentation accuracy on the RL agent's decision-making. The experiment combined a lower-accuracy semantic segmentation model with the decision agent, with the setup being identical to that of the PPO-SS and DDQN-SS environments. The experimental results are shown in Figure 10a,b. The results indicate that PPO-SS consistently achieved higher reward values than PPO-SS(d) with lower segmentation accuracy throughout the training process, suggesting that a decrease in segmentation model accuracy impacts the decision performance of the proposed framework. The reward value increase for PPO-SS(d) was slower compared to PPO-SS, indicating that lower segmentation accuracy interfered with the decision process. Similar to PPO-SS, DDQN-SS had higher overall rewards than DDQN-SS(d), and although DDQN-SS(d) improved in the later stages, it did not reach the performance level achieved with higher segmentation accuracy.



(**a**) (**b**)

Figure 10. Comparison of reward values between PPO-SS(d), DQN-SS(d) with low semantic segmentation accuracy and PPO-SS, DDQN-SS. (**a**) Reward Values of DDQN-SS and DDQN-SS(d). (**b**) Reward Values of PPO-SS and PPO-SS(d).

The comparison of results clearly shows a significant correlation between segmentation accuracy and the RL decision-control framework, with lower segmentation accuracy leading to a decrease in the quality of decision-making outcomes.

Based on the reward values observed in the experiments, particularly for DDQN-SS and PPO-SS, the results showed that, compared to traditional algorithms (DDQN and PPO), the reward trajectory during the training process exhibited relatively smaller fluctuations. This indicated that, even in complex and dynamic traffic scenarios, the framework enhanced by semantic segmentation led to more stable decision-making behavior. The smaller

the standard deviation of the reward values, the higher the reliability and robustness of the method, especially in complex environments with varying traffic density, pedestrian interactions, and dynamic road conditions.

The performance of DDQN-SS and PPO-SS at different stages of the training process demonstrated the adaptability of the framework. As training progressed, the reward values for DDQN-SS and PPO-SS increased at a more stable rate, highlighting the model's ability to adapt to various traffic conditions, road complexities, and environmental changes. This adaptability was further reflected in the model's performance when handling different levels of traffic density and challenging driving scenarios, such as intersections.

By providing higher and more stable rewards during the training process, semantic segmentation enhanced the environmental perception and decision-making capability of the reinforcement learning agent. Reward values, as a key indicator, showed that the integrated approach contributed to achieving more precise and stable decision-control, enhancing the framework's reliability, robustness, and adaptability to real-world driving conditions.

4. Discussion and Future Works

While the proposed framework shows improved decision-control performance in simulated autonomous driving tasks, it faces several limitations. The DeepLabv3P segmentation model, although precise, introduces computational overhead that may limit its real-time applicability, especially on embedded systems. The combined pipeline of semantic segmentation and RL also introduces inference delays, which can affect decision-making in time-critical scenarios. The semantic segmentation network is primarily trained on synthetic CARLA data, which, while diverse, lack the noise and variability of real-world environments. This domain gap may reduce the model's effectiveness when deployed in real-world settings. Furthermore, the framework's ability to handle multi-lane changes and complex urban intersections has not been widely tested. Scenarios like highway merging or multi-lane roundabouts with dynamic traffic signals remain challenging.

Although CNNs show high performance and generalization in semantic segmentation, their explainability and application-specific evaluation are crucial for ensuring reliable decision-making in autonomous systems. Future work will focus on using visualization techniques, such as Grad-CAM, to analyze the internal decision-making process of the CNN and assess its robustness against noise and adversarial disturbances in input images. Experiments will be conducted to evaluate the performance of the semantic segmentation network in various scenarios, including highway driving and adverse weather conditions, to ensure its adaptability and reliability. Furthermore, the impact of segmentation errors on RL decisions will be analyzed, and mechanisms to mitigate their effect on the overall decision-control framework will be proposed.

In this study, semantic segmentation and deep reinforcement learning (DRL) were integrated to enhance decision-control performance in autonomous driving. PPO (Proximal Policy Optimization) and DDQN (Double Deep Q-Network) were chosen as the DRL algorithms due to their simplicity, robustness, and proven effectiveness in simulation-based tasks. While algorithms like SAC (Soft Actor-Critic) and TD3 (Twin Delayed Deep Deterministic Policy Gradient) offer State-of-the-Art performance in continuous action spaces, they are computationally more demanding and were not the primary focus of this work. The choice of PPO and DDQN aligns with the study's objective of demonstrating the impact of semantic segmentation on decision-making performance. Both algorithms serve as strong baselines, allowing for a clear comparison of results with and without segmentation integration. Furthermore, PPO's policy stability and DDQN's suitability for

discrete action spaces made them ideal for the experimental setup, where tasks involved categorical decisions such as lane-keeping and turning.

The simulation results demonstrate the effectiveness of the proposed framework in improving autonomous driving decision-making control. However, simulation-based evaluation has inherent limitations. Simulators like CARLA provide a safe and cost-effective way to test algorithms across various scenarios, but they may not fully capture the complexity and uncertainty of real-world driving conditions. For example, sensor data in real environments often contain noise and distortion, and the predictability of traffic participants' behavior is lower than in simulated models. Additionally, the performance of the framework relies on the accuracy of the semantic segmentation network, which may degrade under conditions like occlusion or adverse weather. The computational overhead of combining semantic segmentation with reinforcement learning (RL) may limit its deployment on resource-constrained platforms. Future work will address these limitations by conducting real-world experiments to validate the framework's generalizability, improve robustness to sensor noise and environmental uncertainty, and optimize computational efficiency for real-time applications.

In the future, we plan to explore some potential improvements:

(1) Expanding the framework to real-world experiments will allow the proposed method to validate the results under more variable and unpredictable conditions, further narrowing the domain gap.
(2) To better handle the differences between simulated and real-world data, such as introducing noise, weather changes, and sensor misalignment during training, domain adaptation techniques could be integrated to improve the model's robustness.
(3) Future work will explore ways to reduce the computational overhead of semantic segmentation without affecting its accuracy. Techniques such as model pruning, quantization, or exploring lightweight segmentation models could be investigated to enhance the system's efficiency for deployment on embedded platforms.

In the extrapolated ideas above, by reducing dependence on raw sensor data (such as RGB images or radar), the semantic segmentation network helps mitigate the impact of noisy or incomplete sensor data, ensuring more stable decisions and fewer performance fluctuations. Additionally, by integrating more advanced techniques, such as domain adaptation, the model can better handle the differences between simulated and real-world environments. This may include considering sensor misalignment, environmental noise, and changing weather conditions, which are typical in real-world scenarios but may be absent or less variable in simulated training data.

5. Conclusions

This paper proposed a decision-control method to enable autonomous vehicles to independently determine driving paths and achieve accurate and stable decision-control. The framework of the proposed method integrated a semantic segmentation neural network with intelligent agent decision models constructed using DDQN and PPO algorithms. Semantic segmentation technology was introduced into the framework to accurately identify object edges and positions, determine the distances and spatial relationships between the vehicle and other objects, and distinguish between drivable and non-drivable areas, providing more realistic road condition references for autonomous driving decisions. Additionally, to improve the convergence speed and training accuracy of the agent model, a reward mechanism suitable for DRL was designed in this study, and the effectiveness of the decision-control method and model was validated through simulation training.

The main conclusions of this study were as follows:

(1) The experimental results conducted on the CARLA simulation platform demonstrated that, in complex urban driving scenarios, the proposed framework outperformed traditional algorithms in terms of stability and decision accuracy, highlighting the advantages of combining semantic understanding with reinforcement learning strategies. Moreover, the proposed DRL reward mechanism further accelerated convergence and ensured high-quality trajectory control.

(2) This study also provided a scalable and modular architecture that could be extended to various traffic environments and real-world applications, thus contributing to the broader field of autonomous driving. As shown in this work, the synergy between perception and decision-making emphasized the importance of integrated approaches in improving the safety, reliability, and efficiency of autonomous systems.

Author Contributions: Conceptualization, N.L.; Methodology, J.G., Z.L. and N.L.; Software, N.L. and J.G.; Validation, N.L. and Z.L.; Investigation, J.G., Z.L., N.L. and Y.G.; Data curation, N.L.; Writing—original draft, N.L. and Z.L.; Writing—review and editing, N.L., Z.L., H.L. and C.X.; Visualization, N.L.; Supervision, Z.L. and J.G.; Project administration, Z.L.; Funding acquisition, Z.L. and J.G. All authors have read and agreed to the published version of the manuscript.

Funding: The project is supported by the National Natural Science Foundation of China (U22A2069), Key R&D Special Project of Henan Province (241111242400-03).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed at the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kalra, N.; Paddock, S.M. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transp. Res. Part A Policy Pract.* **2016**, *94*, 182–193. [CrossRef]
2. Yu, Z.; Jian, W.; Fanqiang, M.; Liu, T. Review on Functional Testing Scenario Library Generation for Connected and Automated Vehicles. *Sensors* **2022**, *22*, 7735. [CrossRef] [PubMed]
3. Papadeas, I.; Tsochatzidis, L.; Amanatiadis, A.; Pratikakis, I. Real-time semantic image segmentation with deep learning for autonomous driving: A survey. *Appl. Sci.* **2021**, *11*, 8802. [CrossRef]
4. Fan, Z.; Liu, K.; Hou, J.; Yan, F.; Zang, Q. JAUNet: A U-shape network with jump attention for semantic segmentation of road scenes. *Appl. Sci.* **2023**, *13*, 1493. [CrossRef]
5. Li, Y.; Shi, J.; Li, Y. Real-Time Semantic Understanding and Segmentation of Urban Scenes for Vehicle Visual Sensors by Optimized DCNN Algorithm. *Appl. Sci.* **2022**, *12*, 7811. [CrossRef]
6. Zhu, Y.; Sapra, K.; Reda, F.A.; Shih, K.J.; Newsam, S.; Tao, A.; Catanzaro, B. Improving semantic segmentation via video propagation and label relaxation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 8856–8865.
7. Liu, C.; Gao, H.; Chen, A. A real-time semantic segmentation algorithm based on improved lightweight network. In Proceedings of the 2020 International Symposium on Autonomous Systems (ISAS), Guangzhou, China, 6–8 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 249–253.
8. Kim, D.; Kwon, J.; Nam, H. End-to-end learning-based self-driving control imitating human driving. In Proceedings of the 2021 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 20–22 October 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1763–1765.
9. Zhu, M.; Wang, Y.; Pu, Z.; Hu, J.; Wang, X.; Ke, R. Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving. *Transp. Res. Part C Emerg. Technol.* **2020**, *117*, 102662.1–102662.14. [CrossRef]
10. Zhu, M.; Wang, X.; Wang, Y. Human-like autonomous car-following model with deep reinforcement learning. *Transp. Res. Part C Emerg. Technol.* **2018**, *97*, 348–368. [CrossRef]

11. Guo, X.; Gao, F.; Zheng, X.; Ning, S. Multi-Agent Collaborative Behavior Decision-Making based on Deep Reinforcement Learning. In Proceedings of the 2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Rio de Janeiro, Brazil, 24–26 May 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 577–581.
12. Chang, C.-C.; Tsai, J.; Lin, J.-H.; Ooi, Y.-M. Autonomous Driving Control Using the DDPG and RDPG Algorithms. *Appl. Sci.* **2021**, *11*, 10659–10675. [CrossRef]
13. Agarwal, T.; Arora, H.; Schneider, J. Learning urban driving policies using deep reinforcement learning. In Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, IN, USA, 19–22 September 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 607–614.
14. Xu, T.; Meng, Z.; Lu, W.; Tong, Z. End-to-end autonomous driving decision method based on improved TD3 algorithm in complex scenarios. *Sensors* **2024**, *24*, 4962. [CrossRef]
15. Kendall, A.; Hawke, J.; Janz, D.; Mazur, P.; Reda, D.; Allen, J.M.; Lam, V.D.; Bewley, A.; Shah, A. Learning to drive in a day. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 8248–8254.
16. Ronecker, M.P.; Zhu, Y. Deep Q-network based decision making for autonomous driving. In Proceedings of the 2019 3rd International Conference on Robotics and Automation Sciences (ICRAS), Wuhan, China, 1–3 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 154–160.
17. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany 8–14 September 2018; pp. 801–818.
18. Tran, Q.D.; Bae, S.H. Comprehensive automated driving maneuvers under a non-signalized intersection adopting deep reinforcement learning. *Appl. Sci.* **2022**, *12*, 9653. [CrossRef]
19. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30, pp. 2094–2100.
20. Zhang, Y.; Du, F.; Wang, M.; Wang, J.; Hu, Y.; Yu, M.; Zhan, A. A Secure Collision Avoidance Warning Algorithm Based on Environmental Characteristics and Driver Characteristics. In Proceedings of the Cyberspace Safety and Security: 11th International Symposium, CSS 2019, Guangzhou, China, 1–3 December 2019; Proceedings, Part II 11; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 545–552.
21. Silva, I.; Silva, H.; Botelho, F.; Pendão, C. Realistic 3D simulators for automotive: A review of main applications and features. *Sensors* **2024**, *24*, 5880. [CrossRef]
22. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.