

Optimizing traffic lights at unbalanced intersections using deep reinforcement learning

Duman Care Khrisne^{1,2}, Made Sudarma², Ida Ayu Dwi Giriantari², Dewa Made Wiharta²

¹Doctoral Program of Engineering Science, Faculty of Engineering, Udayana University, Bali, Indonesia

²Department of Electrical Engineering, Faculty of Engineering, Udayana University, Bali, Indonesia

Article Info

Article history:

Received Jan 23, 2025

Revised Jun 10, 2025

Accepted Jul 10, 2025

Keywords:

Deep reinforcement learning

Optimize

Simulation

Traffic signal

Unbalanced traffic

Waiting time

ABSTRACT

Unbalanced intersectional traffic flow increases vehicle delays, fuel consumption, and pollution. This study investigates the application of deep reinforcement learning (DRL) to optimize traffic signal timing at the Pamelisan intersection in Denpasar, Indonesia. Real-world traffic data were incorporated into a SUMO microsimulation environment to train DRL agents using the deep Q-network (DQN) algorithm. Experimental results show that DRL-based optimization reduced the average vehicle waiting time from 594.49 seconds (static control) to 169.44 seconds and 173.10 seconds for agents trained without and with noise, respectively. The average vehicle speed remained stable at 5.6–5.97 m/s across all scenarios, indicating enhanced traffic efficiency without adverse effects. The findings underscore the effectiveness and adaptability of DRL in addressing traffic inefficiencies, optimizing them, and offering a robust solution for dynamic traffic management at unbalanced traffic intersections in urban areas.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Duman Care Khrisne

Doctoral Program of Engineering Science, Faculty of Engineering, Udayana University

Bali, Indonesia

Email: duman@unud.ac.id

1. INTRODUCTION

Transportation problems are still often caused by traffic congestion, which has an impact on traffic accidents, pollution, and economic losses [1], [2]. Previous research has succeeded in summarizing several techniques that can be used to solve traffic problems. Previous research has succeeded in summarizing several techniques for solving traffic problems based on their completion time [3]. These techniques are grouped into long-term, medium-term, and short-term solutions. One of the short-term or real-time techniques is carried out through good management of traffic flow at intersections [4], [5]. A traffic light system can manage short-term traffic flow at intersections [5], [6]. Adaptive signal control methods, such as split, cycle and offset optimization technique (SCOOT) [7] and Sydney coordinated adaptive traffic (SCAT) [8] are widely used in traffic light management systems. They mostly rely on manually scheduled signal phases and work well when traffic flow is nearly equal in all directions. This schedule changes dynamically by looking only at traffic volume using induction loop sensors. As a result, signals cannot see and react to changes in traffic patterns in real time, and transportation operators often have to manually change signal phases to keep up with traffic conditions [9]. Furthermore, it is often possible to find more traffic in one direction than the other (unbalanced traffic flow).

The traditional system lacks intelligent management, which results in people waiting, regardless of the absence of vehicles from the opposite direction. This inevitable waiting time sometimes makes people restless, often ending in violation of rules and accidents [10]. Furthermore, this leads to more fuel consumption

and pollutes the surrounding environment. Therefore, intelligent or dynamic traffic light control needs to be continuously improved, especially when looking at other factors, such as waiting time at intersections and the traffic volume factor, which is currently widely used. Artificial intelligence, which has developed in this decade, provides hope for the emergence of systems with high intelligence and adaptation. Research [11]–[15] uses reinforcement learning (RL) and deep reinforcement learning (DRL) approaches to provide solutions to overcome traffic congestion. The RL model directly tries to adapt to solve the problems imposed on the model, including traffic congestion problems. Despite its success, RL still has shortcomings. When dealing with state-action spaces that are too large, RL algorithms often require manual division of space into smaller and separate parts to represent different states (state-action space discretization). Discretization of the action-state space causes the complexity of the problems RL can solve to be limited and time-consuming [16].

DRL comes as a development of conventional RL by adding a deep neural network (DNN) to RL. DRL has succeeded in overcoming several weaknesses of conventional RL. One of the main reasons is that DRL uses DNN, which can overcome higher problem complexity and represent more complex value functions or policies [17]. Thus, DRL can address problems with larger dimensions and more complex environments, which are difficult to handle by conventional RL. In addition, DNN in DRL can automatically learn more meaningful feature representations from input data, allowing agents to recognize more complex patterns and make better decisions [18]–[20]. DRL, which has the advantage of handling large-scale and high-complexity problems, makes it an attractive choice for covering the weaknesses of conventional RL in research to build a more adaptive traffic light control system.

A traffic simulator is often used to evaluate traffic control strategies [21], emphasizing sustainability, safety, and traffic efficiency performance indicators. Researchers have used two main methods to test traffic simulators: macroscopic and microscopic. Several studies have used macroscopic simulations to mimic real-world traffic dynamics [22], [23]. However, more and more studies are turning to microscopic simulations, such as SUMO, VISSIM, and AIMSUN, which offer a more comprehensive depiction of complex traffic dynamics, including the stochastic character of driving and route choices [21]. SUMO, as one of the microscopic simulators, is widely used to evaluate traffic control strategies. However, to the authors' knowledge, no SUMO simulation has been built using actual traffic flow data (and real-world road networks) to demonstrate the unbalanced traffic flow state.

Furthermore, according to Tan *et al.* [18], most DRL work is still not ready for direct application in real-world traffic because, until now, the DRL agent is assumed to have perfect knowledge of the traffic environment. In reality, a congestion detection or prediction system is highly desired to estimate traffic conditions with significant disturbances, one of which is unbalanced traffic flow. Therefore, in this study, an adaptive traffic control system was built using deep Q-network (DQN) [24]. This DRL algorithm is used to optimize vehicle waiting time at signalized intersections by optimizing changes in traffic light times. The DQN in this study was trained using SUMO microscopic simulation data with characteristics of unbalanced traffic flow and perturbation of queue length.

2. METHOD

This research is built with four main steps. First, the number of vehicles passing through one lane at an intersection was calculated using YOLOv8. Next, an intersection simulation will be built using the previous calculation data using SUMO microsimulation, which will be continued by training DRL agents using SUMO simulation as input. Finally, the optimization that the DRL agent did will be analyzed. Figure 1 shows us the research flow diagram.

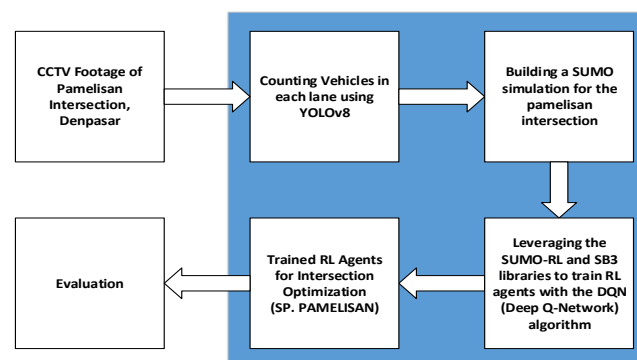


Figure 1. Research stages flow

2.1. Calculating vehicle flow

The data used to build the intersection simulation with unbalanced traffic flow was taken from the Dinas Perhubungan Provinsi Bali or Bali Provincial Transportation Agency, which utilized surveillance cameras installed at the Pamelisan intersection. The data obtained is a video recording of the surveillance camera. The video is used as input to calculate vehicles passing through the lanes at the Pamelisan intersection. The configuration of the position of the vehicle flow counter-point on the lane at the Pamelisan intersection is shown in Figure 2. Figure 2 shows that six counter-points were used to calculate vehicle flow in each lane at the intersection. Table 1 presents a more concise relationship between the position of the point and the direction of vehicle flow calculated at each counter-point. To count vehicles, we use the YOLOv8 object detection algorithm [25]. YOLOv8 is used to detect vehicles passing through a lane. After that, we use an object tracking and counting algorithm made explicitly for tracking objects from the results of YOLOv8 detection to count the traffic flow. The flowchart of the tracking and counting algorithm is shown in Figure 3.

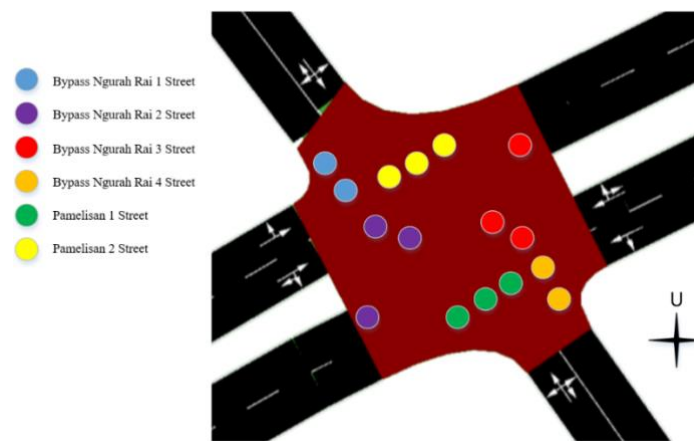


Figure 2. Configuration of the vehicle flow counter point

Table 1. Summary of vehicle flow direction at each counting point

Label	Counting point	Vehicle flow direction
●	Ngurah Rai 1	Bypass Ngurah Rai → Pamelisan (East-North); Bypass Ngurah Rai → Bypass Ngurah Rai (East-West)
●	Ngurah Rai 2	Bypass Ngurah Rai → Bypass Ngurah Rai (East-West); Bypass Ngurah Rai → Pamelisan (East-South) & Bypass Ngurah Rai → Bypass Ngurah Rai (East-East/U-turn)
●	Ngurah Rai 3	Bypass Ngurah Rai → Bypass Ngurah Rai (West-East); Bypass Ngurah Rai → Pamelisan (West-North) & Bypass Ngurah Rai → Bypass Ngurah Rai (West-West/U-turn)
●	Ngurah Rai 4	Bypass Ngurah Rai → Pamelisan (West-South) & Bypass Ngurah Rai → Bypass Ngurah Rai (West-East)
●	Pamelisan 1	Pamelisan → Baypass Ngurah Rai (South-East), Pamelisan → Baypass Ngurah Rai (South-West); Pamelisan → Pamelisan (South-North)
●	Pamelisan 2	Pamelisan → Baypass Ngurah Rai (North-East), Pamelisan → Baypass Ngurah Rai (North-West); Pamelisan → Pamelisan (North-South)

2.2. Building a SUMO simulation

After obtaining the vehicle flow data that passes through each road lane at the Pamelisan intersection, the next step is translating the vehicle flow into the SUMO microsimulation [26]. In SUMO, the vehicle flow is converted into a vehicle emergence simulation using the routes function. SUMO is a well-known open-source traffic simulator that provides practical graphical user interfaces (GUIs) and application programming interfaces (APIs) for efficiently managing and modeling road networks. It offers a visual graphical interface for creating different road network architectures in many grid formats and allows dynamic routing [16]. Additionally, SUMO supports OpenStreetMap (OSM). A full scenario may be created quickly and easily with the help of the OSM script. Typemaps and settings appropriate for the chosen traffic modes will be imported into the network. Furthermore, SUMO can control each intersection's traffic lights using user-defined policies. SUMO makes it possible to take pictures at every simulation stage, giving us the state data for our study. SUMO simulation for the Pamelisan intersection was built with data calculated from the actual traffic flow obtained in the previous step. Using this data, we build a simulated environment that imitates real-world traffic flow at the Pamelisan intersection, which has an unbalanced traffic flow and static traffic light phase.

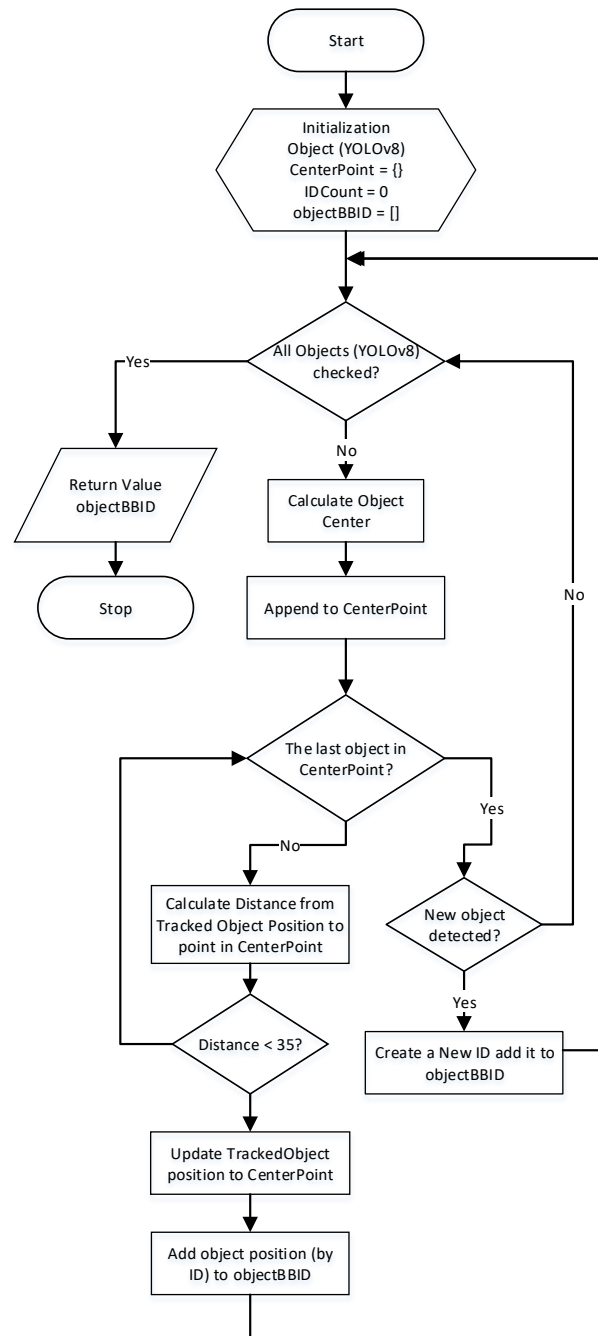


Figure 3. Flowchart of tracking object algorithm

2.3. Deep reinforcement learning agent training

In this study, the DRL agent was built using the SUMO-RL library [27]. SUMO-RL provides a simple interface to create a RL environment with SUMO for traffic signal control. The DRL agent built is an agent that uses the DQN algorithm for the training process to optimize vehicle waiting time at the Pamelisan intersection. DQN in SUMO-RL was built using the stable baselines3 (SB3) library [28].

DQN given input as a simulation generated in the previous stage as an environment (SUMO environment). Because this study only optimizes one traffic light (Pamelisan intersection), the DRL agent used is one (single agent). The agent performs optimization using the Markov decision process (MDP) model with three components: observation, action, and reward.

For observation space, DQN uses DNN, which has 15 inputs generated from observations in the environment, namely two green phases (north-south and east-west green lights), one transition phase

(yellow light), and density and queue values on six lanes of the road at the Pamelisan intersection (12 inputs). In addition to standard observations in training, observations with noise were made, which were carried out by changing the queue length value. This is a way to represent real-world conditions where large vehicles sometimes block surveillance cameras, which causes changes in the queue length value at intersections. Action space generated as output from DNN. There are 2 action spaces in this study, as illustrated in Figure 4. One for Ngurah Rai road (East-West), depicted in Figure 4(a), and one for the green phase at Pamelisan road (North-South), depicted in Figure 4(b). The reward function is calculated using changes in cumulative vehicle delay. In other words, the reward is how much the total delay (the sum of the waiting times of all approaching vehicles) changes concerning the previous time step.

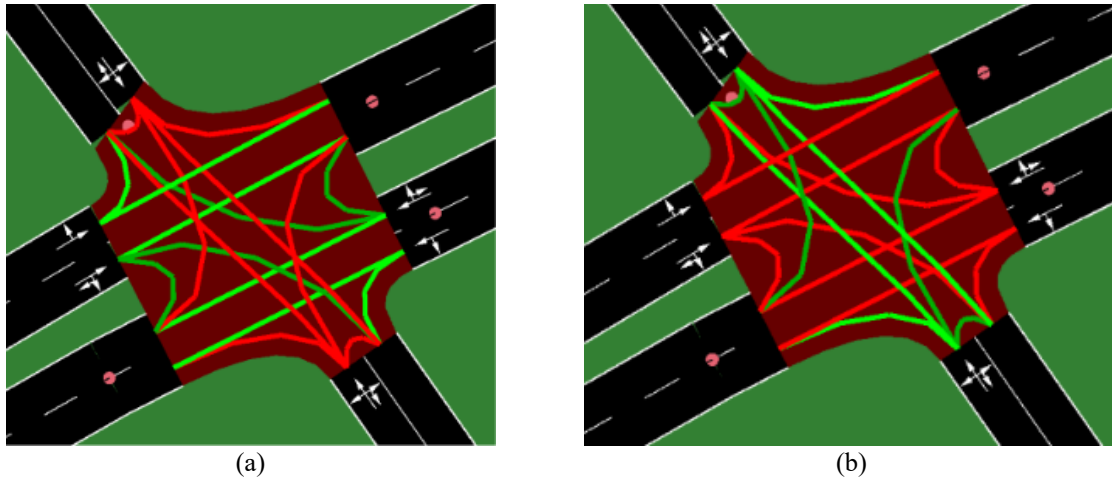


Figure 4. Pamelisan intersection in SUMO with (a) Ngurah Rai green phase and (b) Pamelisan green phase

2.4. Evaluation

The final stage of this research is the evaluation process carried out on the DRL model, which was trained in the previous stage. Two evaluation metrics were used to measure the agent's optimization ability: the accumulated waiting time at the intersection (for all road lanes) and the average vehicle speed at the intersection. Vehicle waiting time is defined as the time (in seconds) spent below 0.1 m/s since the last time the vehicle traveled faster than 0.1 m/s. (The vehicle waiting time is reset to 0 every time the vehicle moves). In (1) calculates the total vehicle waiting time at an intersection.

$$total_W = \sum_{i=1}^L \sum_{j=1}^{V_i} W_{ij} \quad (1)$$

Where $total_W$ is the total waiting time in all lanes of the road in the intersection, L is the number of lanes of the road in the intersection, V_i is the number of vehicles in lane i and W_{ij} is the waiting time of vehicle j in lane i .

The average vehicle speed at an intersection is calculated by finding the average speed of the vehicles at the intersection, normalized by the maximum speed allowed for each vehicle. If there are no vehicles at the intersection, this function returns 1. In (2) calculates the normalized average vehicle speed at an intersection.

$$avg_speed = \frac{1}{N} \sum_{i=1}^N \frac{S_i}{S_{max,i}} \quad (2)$$

Where avg_speed is the normalized average speed for all vehicles at the intersection, N is the number of vehicles at the intersection, S_i is the speed of vehicle i at the time of observation, and $S_{max,i}$ is the maximum speed permitted for vehicle i (in the simulation each type of vehicle is set to a maximum permitted speed).

To evaluate the agent's optimization capability, each average waiting time and average speed produced after training results are compared with the initial simulation data's average waiting time and speed. Since the initial simulation data used is static traffic lights, comparing the average waiting time and speed before and after training can show the changes in the average waiting time and speed at the intersection when using static time and the DRL agent. We also did another evaluation to see whether DRL agents have some different results when faced with training with noise. This evaluation will compare DRL agents trained using noisy data (perturbation) to agents that do not experience noise in their training.

3. RESULTS AND DISCUSSION

3.1. Vehicle flow counting result

Video data obtained from Bali Provincial Transportation Agency, produced from surveillance cameras at the Pamelisan intersection. Because there are no surveillance cameras with a wide view at the Pamelisan intersection, and only pan-tilt-zoom (PTZ) cameras are available, data for the entire lane cannot be taken at once. The video was taken for 4 days from January 7th, 2024 to January 11th, 2024, to obtain video recordings of vehicles passing through each lane at the Pamelisan intersection in Denpasar, one road arm for one day of recording. For 4 days, recordings were obtained from each road arm and each lane of the road could be calculated. In this study, to represent the conditions and average density of each lane, video recordings were selected for 7,200 seconds (120 minutes) at the same time on different days (recordings of one arm and the other were on different days). The selected video time was 13.00 to 15.00 WITA.

The obtained video calculated vehicle flow using YOLOv8 by calculating vehicles passing through the perimeter box configured as in Figure 2. The vehicle classes that are set to be recognized by YOLOv8 are cars, buses, and trucks. In this study, we did not calculate the flow of 2-wheeled vehicles (motorcycles). As a result, 1,504 vehicles passed through Ngurah Rai 1 and Ngurah Rai 2 lanes, 1,013 vehicles passed through Ngurah Rai 3 and Ngurah Rai 4 lanes, 1,017 vehicles passed through Pamelisan 2 and 236 vehicles passed through Pamelisan 1 lanes. Table 2 presents the number of vehicle flows (based on observation position) and the destination of vehicles passing through the Pamelisan intersection. It should be noted that observations were not carried out at once, so for Ngurah Rai 1 and Ngurah Rai 2 observations were carried out on the first day, Ngurah Rai 3 and Ngurah Rai 4 on the second day, Pamelisan 1 on the third day and the fourth day for Pamelisan 2.

Table 2. Vehicle flow counting result

Depart lane	Arrival lane	Vehicle count	Total
Pamelisan 1 (South)	Ngurah Rai 1 (East)	48	236
	Ngurah Rai 2 (East)	44	
	Ngurah Rai 4 (West)	77	
	Pamelisan 1 (North)	67	
Pamelisan 2 (North)	Ngurah Rai 1 (East)	234	1017
	Ngurah Rai 1 (East) [#]	2	
	Ngurah Rai 3 (West)	687	
	Ngurah Rai 3 (West) [#]	23	
	Pamelisan 2 (South)	71	
Ngurah Rai 3 (East)*	Ngurah Rai 3 (West)	39	289
	Pamelisan 1 (North)	250	
Ngurah Rai 4 (East)*	Ngurah Rai 4 (West)	652	724
	Pamelisan 2 (South)	72	
Ngurah Rai 2 (West)*	Ngurah Rai 2 (East)	247	324
	Pamelisan 2 (South)	77	
Ngurah Rai 1 (West)*	Ngurah Rai 1 (East)	418	1180
	Pamelisan 1 (North)	762	

[#] Lane changing occur

* As the position of departing is close together, we assume the departing point from arrival lane

3.2. SUMO simulation

This paper presents a case study of real-world traffic at Pamelisan intersection, the intersection of Pamelisan Road and Ngurah Rai Road in Denpasar City. We simulated the Pamelisan intersection using the SUMO simulator. The intersection layout in SUMO is depicted in Figure 4. Pamelisan intersection features four directions, and Ngurah Rai Road has two lanes. The leftmost lane in Ngurah Rai is designated for left turns and going straight, and the rightmost lane is reserved for right turns, going straight, and u-turn. Pamelisan Road has one lane with no designated vehicle turns (free for all turns). In addition to building road lanes, SUMO simulation also simulates vehicle flows, as described in Table 2.

The Pamelisan intersection that we simulated using SUMO has two green light phases and two transition phases. One green light phase is for Ngurah Rai Road, depicted in Figure 4(a), and one for Pamelisan Road, depicted in Figure 4(b). These two green phases also work as action state for DQN. Each phase has a transition phase (yellow light) that cycle between green and yellow phase. These green and yellow phases have their duration and are called signal phase duration. For the Pamelisan intersection, the standard (fix-timed) signal phase duration is shown in Table 3.

Table 3. Pamelisan intersection fix-timed signal phase duration

Phase	Duration (second)	Cycle length (second)
Ngurah Rai green phase	39	90
Ngurah Rai yellow phase	6	
Pamelisan green phase	39	
Pamelisan yellow phase	6	

3.3. Agent training

We perform agent training using multi-layer perceptron (MLP) policy from SB3, hyperparameters for the model can be seen in Table 4. We perform training using 72,000 timesteps on two observation mode settings. The first setting is using observations without noise and the second setting is using observations with noise. The setting without noise undertakes the vehicle flow value in Table 2 without changes. The setting with noise perturbation is carried out by inserting noise into the length of the vehicle queue at the intersection so that there is a change in the observation. The inserted noise is generated randomly using the random Gaussian function, with mean =0 and standard deviation =1.

Table 4. Hyperparameter value for DQN model

Parameter	Value
Learning rate	1e-3
Learning start	5
Training frequency	4
Gamma	0.9
Exploration fraction	0.1
Exploration final episode	0.05
Target update interval	500
Replay buffer size	50000

After training for 50 episodes, the average reward obtained by the agent per episode is summarized in Figure 5. Figure 5(a) shows the average reward obtained by the agent when trained in a setting without noise, at the beginning of training the reward obtained is relatively small but as the training episodes increase, higher rewards are obtained and tend to be stable. Figure 5(b) shows the average reward obtained by the agent through training with a setting adding noise. Since the beginning of training, the reward obtained by the agent is more stable in this setting. Stable agent reward during agent training with noise because the agent learns not to monitor the queue length in the environment too much at the beginning of training. Although there is a difference initially, both agents get stable rewards at the end of training. This reward shows that the agent has learned how to optimize at this intersection (with an unbalanced traffic flow). Evidence for this statement can be seen in Figure 6, which shows that the average waiting time at the intersection between agents looks close together.

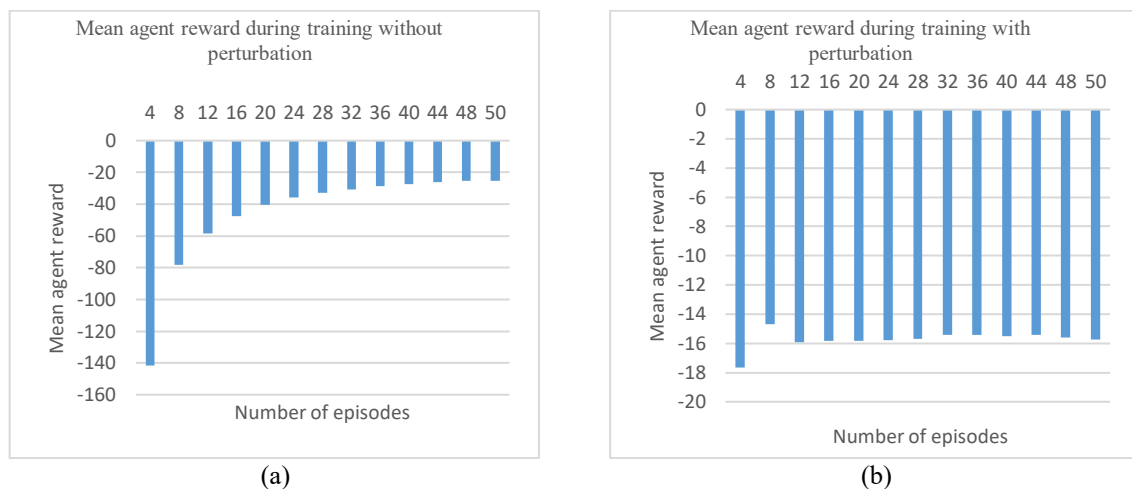


Figure 5. Agent reward while training (a) without noise perturbation and (b) with perturbation

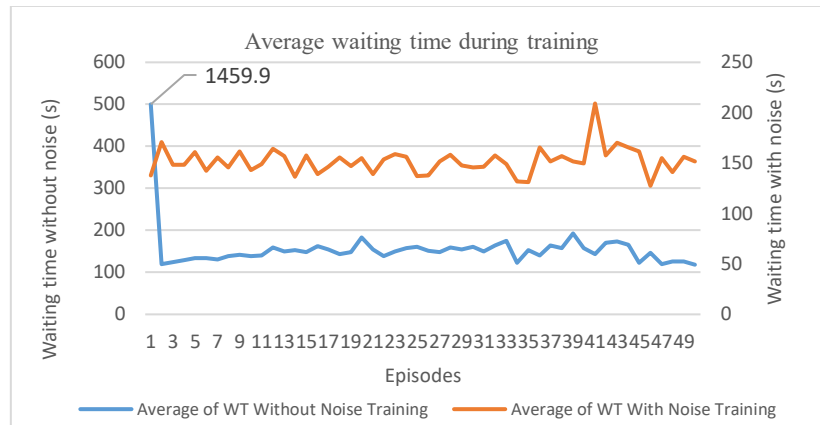


Figure 6. Intersection average waiting time during training without noise perturbation and training with perturbation

Training history in Figure 6 also shows that the agent trained without using noise at the beginning of training has difficulty finding a generalization pattern of waiting time to queue length. This difficulty causing the agent to explore at the beginning of training. After getting the exploration pattern, the agent shows quite good exploitation, so that at the end of training the waiting time at the intersection is even better than the agent trained using noise. On the other hand, the agent trained using noise has a much smaller waiting time at the beginning of training, but as training progresses, the waiting time value tends to remain the same. This shows that if we want faster exploitation, we can use training with noise, but if we want a higher waiting time reward, we can use training without noise.

3.4. Evaluating optimization result

Evaluation of the optimization performed by the DRL agent at the Pamelisan intersection is measured using SUMO simulation and built using an unbalanced traffic flow. Furthermore, the simulation results of the intersection optimized using the DRL agent were compared with those of the intersection simulation without optimization. Testing was also conducted using a simulation containing noise to see whether the agent could overcome noise at the intersection. The noise used during the evaluation was generated in the same way as during training.

Agents were evaluated on one simulation episode with noise and one without noise (both with imbalanced flows). This simulation was introduced to agents trained with noise and agents trained without noise. Figure 7 shows the evaluation results of each agent for optimization with waiting time as the measured parameter. Figure 7(a) depicts the waiting time value before agent optimization, with an average waiting time of 594.49 seconds at each intersection. When optimizing a simulation with noise, Figure 7(b) depicts the optimization outcomes obtained by an agent trained without noise during its training period. Figure 7(c) depicts the optimization results achieved by an agent trained on noisy data during its training phase. When optimization is performed on a noise-free simulation, Figure 7(d) displays the optimization results of an agent trained without noise during its training period. Figure 7(e) displays the optimization outcomes of an agent trained with noisy data during its training period.

According to Figure 7, the average waiting time at the intersection is lower for Figures 7(b) to 7(e), (248.5, 173.1, 169.4, and 186.5 seconds respectively) than for the intersection without optimization Figure 7(a). Furthermore; we can compare Figures 7(b) and 7(c) to see the agent's optimization ability when faced with a simulation with noise on the queue length. From these two images, we can see that the agent trained with noise on the training data in Figure 7(c), can become accustomed to optimizing at the intersection earlier. Although the agent in Figure 7(b) was similarly successful in optimizing, it is clear that there were multiple instances where the agent became confused by the noise that arose, increasing the average waiting time. Figures 7(d) and 7(e) demonstrate the agent's evaluation in the simulation with no noise on queue length. If the agent had difficulty dealing with noise in the previous evaluation, there was no noise in the simulation this time, so the agent trained without noise Figure 7(d) and with noise Figure 7(e) had no difficulty optimizing; it's just that the agent in evaluation Figure 7(d) appeared to optimize better when measured by its average waiting time. Meanwhile, if we look at it, Figure 7(e) is not much different from Figure 7(d), and the optimization is not much different; it's just that because it is trained using training data containing noise, the agent is more careful and has difficulties at the beginning of the episode, but becomes more accustomed at the end of the evaluation episode.

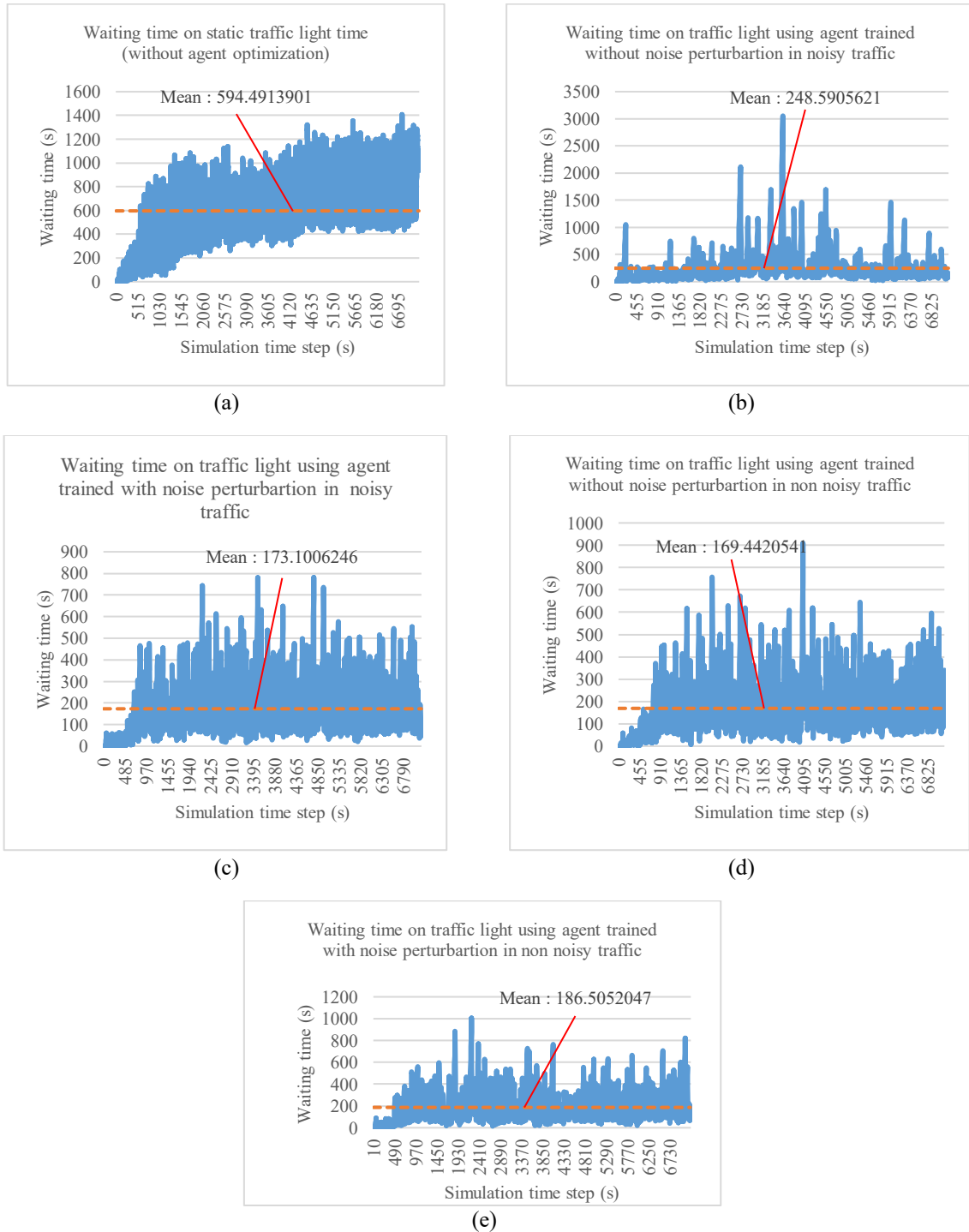


Figure 7. Waiting time at Pamelisan intersection under different strategies: (a) without optimization, (b) using DRL agent trained without noise perturbation, (c) using DRL agent trained with noise perturbation, (d) using DRL agent trained without noise perturbation in non noisy traffic, and (e) using DRL agent trained with noise perturbation in non noisy traffic

Apart from waiting time, evaluation is also done by considering the average speed of vehicles at the intersection. Figure 8 shows the average speed of vehicles when the evaluation simulation is carried out. For this scenario, Figure 8(a) shows when the intersection is not optimized using an agent. Figure 8(b) shows the average speed of vehicles when the intersection is optimized using an agent trained without noise, the intersection simulation is not affected by noise in the queue. Figure 8(c) shows the average speed of vehicles at an intersection optimized using an agent trained using noise in the queue length, the evaluation

intersection simulation is also affected by the same noise as the noise during training. The average speed of cars traveling through the intersections, whether optimized or not, is not significantly different, as shown in Figures 8(a) to 8(c) (5.6-5.9 m/s or 20.16-21.24 km/h). Given the decreases in average waiting time at the intersection, it indicates that optimization has succeeded in increasing the flow of vehicles passing through the intersection without requiring an increase in average vehicle speed. This suggests that the average speed at the intersection does not change much, even with optimization agents present or not.

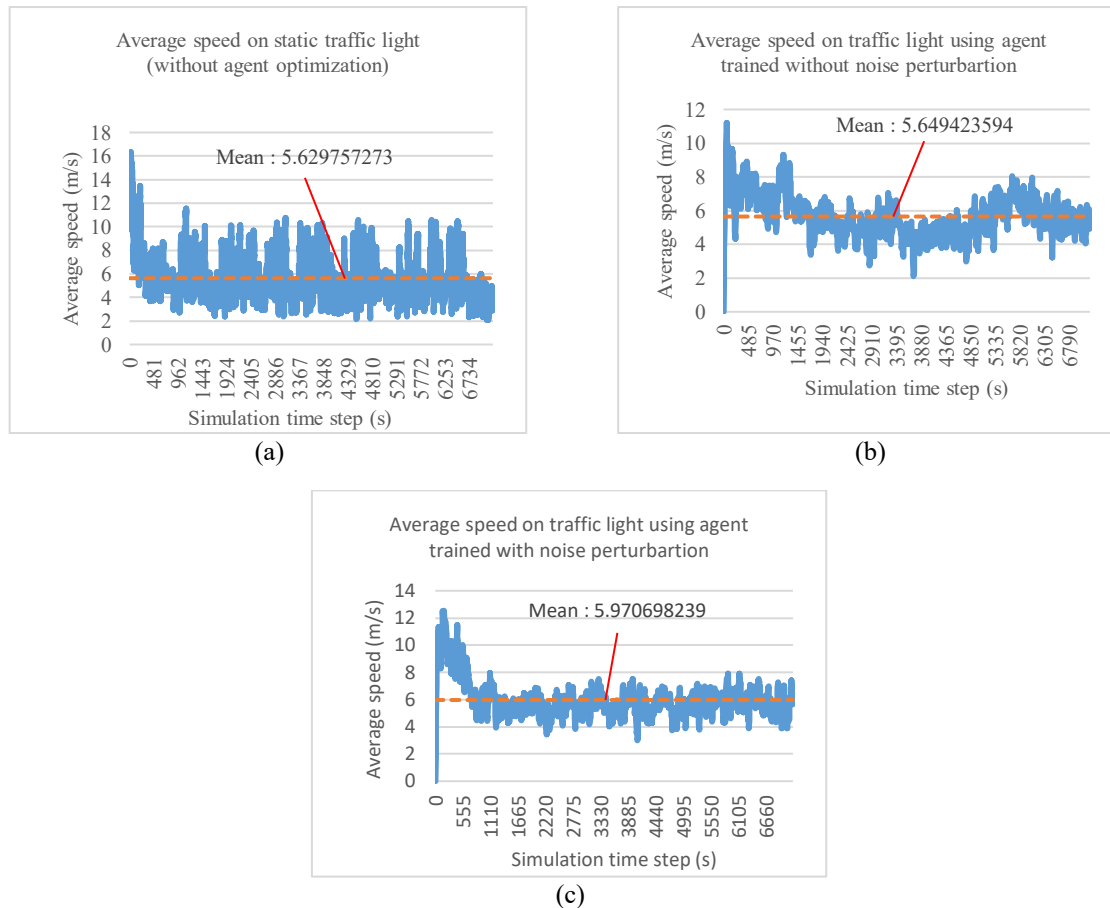


Figure 8. Average speed at intersection with different strategy, (a) without optimization, (b) using DRL agent trained without noise perturbation, and (c) using DRL agent trained with noise perturbation

4. CONCLUSION

This study effectively used the DQN algorithm in conjunction with DRL to optimize traffic signal time at the Pamelisan intersection with unbalanced traffic flow. Vehicle waiting time was significantly reduced, according to experimental data. With the DRL agent trained without noise, the average waiting time dropped from 594.49 seconds in the baseline static traffic signal system to as low as 169.44 seconds, and with the DRL agent trained with noise, it dropped to 173.10 seconds. These enhancements show that intersection efficiency is efficiently optimized by both noise-trained and noise-free DRL agents, with the noise-trained agent demonstrating superior robustness in noisy environments. Furthermore, the average vehicle speed (about 5.6–5.97 m/s) stayed constant across situations, demonstrating that the optimization enhanced traffic flow without sacrificing smoothness or safety. These findings demonstrate how DRL-based systems have the ability to completely transform adaptive traffic control in intricate and ever-changing metropolitan settings. By using DRL-based DQN, the intersection efficiency is significantly improved, demonstrating the potential for implementation in real-world scenarios.

FUNDING INFORMATION

The Faculty of Engineering at Udayana University and the Institute for Research and Community Service Udayana University funding this work with grant number B/255.3/UN14.4.A/PT.01.03/2024.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Duman Care Khrisne	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓			✓
Made Sudarma	✓	✓		✓		✓	✓			✓		✓	✓	
Ida Ayu Dwi Giriantari	✓			✓			✓			✓		✓		
Dewa Made Wiharta	✓				✓			✓		✓		✓		

C : Conceptualization

M : Methodology

So : Software

Va : Validation

Fo : Formal analysis

I : Investigation

R : Resources

D : Data Curation

O : Writing - Original Draft

E : Writing - Review & Editing

Vi : Visualization

Su : Supervision

P : Project administration

Fu : Funding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Derived data supporting the findings of this study are available from the corresponding author, [DCK], on request.

REFERENCES




- [1] M. S. Ali, M. Adnan, S. M. Noman, and S. F. A. Baqueri, "Estimation of traffic congestion cost - a case study of a major arterial in Karachi," *Procedia Engineering*, vol. 77, pp. 37–44, 2014, doi: 10.1016/j.proeng.2014.07.030.
- [2] J. Fan, A. Li, A. Ilahi, and K. Gao, "Emission impacts of left-turn lane on light-heavy-duty mixed traffic in signalized intersections: Optimization and empirical analysis," *Heliyon*, vol. 9, no. 5, 2023, doi: 10.1016/j.heliyon.2023.e16260.
- [3] M. Akhtar and S. Moridpour, "A review of traffic congestion prediction using artificial intelligence," *Journal of Advanced Transportation*, vol. 2021, 2021, doi: 10.1155/2021/8878011.
- [4] M. A. Mondal and Z. Rehena, "Priority-based adaptive traffic signal control system for smart cities," *SN Computer Science*, vol. 3, no. 5, 2022, doi: 10.1007/s42979-022-01316-5.
- [5] D. Zavanis, D. Mandalozis, A. Yasar, and L. Hasimi, "The importance of implementation of traffic system: Greece case study," *Procedia Computer Science*, vol. 238, pp. 610–617, 2024, doi: 10.1016/j.procs.2024.06.068.
- [6] L. Ramirez-Polo, M. A. Jimenez-Barros, V. V. Narváez, and C. P. Daza, "Simulation and optimization of traffic lights for vehicles flow in high traffic areas," *Procedia Computer Science*, vol. 198, pp. 548–553, 2021, doi: 10.1016/j.procs.2021.12.284.
- [7] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and M. C. Royle, "The SCOOT on-line traffic signal optimisation technique," *Traffic Engineering & Control*, vol. 23, no. 4, pp. 190–192, 1978.
- [8] A. G. Sims and K. W. Dobinson, "The Sydney coordinated adaptive traffic (SCAT) system philosophy and benefits," *IEEE Transactions on Vehicular Technology*, vol. 29, no. 2, pp. 130–137, 1980, doi: 10.1109/T-VT.1980.23833.
- [9] D. Garg, M. Chli, and G. Vogiatzis, "Fully-autonomous, vision-based traffic signal control: from simulation to reality," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 2022, pp. 454–462.
- [10] H. Khan et al., "Machine learning driven intelligent and self adaptive system for traffic management in smart cities," *Computing*, vol. 104, no. 5, pp. 1203–1217, 2022, doi: 10.1007/s00607-021-01038-1.
- [11] H. Abdelgawad, B. Abdulhai, S. El-Tantawy, A. Hadayeghi, and B. Zvaniga, "Assessment of self-learning adaptive traffic signal control on congested urban areas: Independent versus coordinated perspectives," *Canadian Journal of Civil Engineering*, vol. 42, no. 6, pp. 353–366, 2015, doi: 10.1139/cjce-2014-0503.
- [12] D. A. Vidhate and P. Kulkarni, "Cooperative multi-agent reinforcement learning models (CMRLM) for intelligent traffic control," in *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*, 2017, pp. 325–331, doi: 10.1109/ICISIM.2017.8122193.
- [13] C. H. Wan and M. C. Hwang, "Value-based deep reinforcement learning for adaptive isolated intersection signal control," *IET Intelligent Transport Systems*, vol. 12, no. 9, pp. 1005–1010, 2018, doi: 10.1049/iet-its.2018.5170.
- [14] N. Bhawe, A. Dhagavkar, K. Dhande, M. Bana, and J. Joshi, "Smart signal - adaptive traffic signal control using reinforcement learning and object detection," in *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2019, pp. 624–628, doi: 10.1109/I-SMAC47947.2019.9032589.
- [15] M. Sahal, Z. Hidayat, Y. Bilfaqih, M. A. Hady, and Y. M. H. Tampubolon, "Smart traffic light using YOLO based camera with deep reinforcement learning algorithm," *JAREE (Journal on Advanced Research in Electrical Engineering)*, vol. 7, no. 1, pp. 13–19, 2023, doi: 10.12962/jaree.v7i1.335.
- [16] T. Pan, "Traffic light control with reinforcement learning," *arXiv-Computer Science*, pp. 1–18, 2023.
- [17] X. Zang, H. Yao, G. Zheng, N. Xu, K. Xu, and Z. Li, "MetaLight: Value-based meta-reinforcement learning for traffic signal control," *AAAI Conference on Artificial Intelligence*, vol. 34, no. 1, pp. 1153–1160, 2020, doi: 10.1609/aaai.v34i01.5467.
- [18] K. L. Tan, A. Sharma, and S. Sarkar, "Robust deep reinforcement learning for traffic signal control," *Journal of Big Data Analytics in Transportation*, vol. 2, no. 3, pp. 263–274, 2020, doi: 10.1007/s42421-020-00029-6.
- [19] Z. Ma, T. Cui, W. Deng, F. Jiang, and L. Zhang, "Adaptive optimization of traffic signal timing via deep reinforcement learning," *Journal of Advanced Transportation*, vol. 2021, 2021, doi: 10.1155/2021/6616702.
- [20] M. Zhu, X.-Y. Liu, S. Borst, and A. Walid, "Deep reinforcement learning for traffic light control in intelligent transportation systems," *arXiv-Computer Science*, pp. 1–17, 2023.

Optimizing traffic lights at unbalanced intersections using deep reinforcement ... (Duman Care Khrisne)




- [21] Y. Han, M. Wang, and L. Leclercq, "Leveraging reinforcement learning for dynamic traffic control: a survey and challenges for field implementation," *Communications in Transportation Research*, vol. 3, 2023, doi: 10.1016/j.commtr.2023.100104.
- [22] V. Pandey, E. Wang, and S. D. Boyles, "Deep reinforcement learning algorithm for dynamic pricing of express lanes with multiple access locations," *Transportation Research Part C: Emerging Technologies*, vol. 119, 2020, doi: 10.1016/j.trc.2020.102715.
- [23] D. Zhou and V. V. Gayah, "Scalable multi-region perimeter metering control for urban networks: a multi-agent deep reinforcement learning approach," *Transportation Research Part C: Emerging Technologies*, vol. 148, 2023, doi: 10.1016/j.trc.2023.104033.
- [24] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv-Computer Science*, pp. 1–9, 2013.
- [25] G. Jocher *et al.*, "YOLO by ultralytics," *GitHub*. 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [26] P. A. Lopez *et al.*, "Microscopic traffic simulation using SUMO," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2575–2582, doi: 10.1109/ITSC.2018.8569938.
- [27] Daraan and L. N. Alegre, "SUMO-RL," *GitHub*. 2019. [Online]. Available: <https://github.com/LucasAlegre/sumo-rl>
- [28] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, pp. 1–8, 2021.

BIOGRAPHIES OF AUTHORS






Duman Care Khrisne    completed Bachelor of Engineering Education in the Department of Electrical Engineering at Udayana University in 2008. In 2012, he received a master's degree in the Master's Program in Electrical Engineering at Udayana University. In 2014, began working as a lecturer in the Electrical Engineering Program at Udayana University, where he taught the computer engineering major. His research field focuses on image processing, pattern recognition, artificial neural networks, deep learning, and reinforcement learning. He also contributes as a book writer and reviewer in journals. He can be contacted at email: duman@unud.ac.id.






Prof. Made Sudarma    has been serving as a lecturer in the Electrical Engineering Study Program, Faculty of Engineering, Udayana University since March 1993. Completed Bachelor of Electrical-Computer Engineering education at the Sepuluh Nopember Institute of Technology, Surabaya in 1992. In 2000 he completed Master of Applied Science (M.A.Sc.) at School of Information Technology and Engineering, Ottawa University Canada. In 2012, he completed his doctoral program at the Postgraduate Program of Udayana University. Since November 2019, has been appointed as a permanent professor in the field of information technology in the Electrical Engineering Study Program, Faculty of Engineering, Udayana University. His research focus on internet and web applications, cloud computing, artificial intelligence, data warehousing and data mining, computer graphics, and virtual reality. He also contributes as a book writer, and as a reviewer in international and national journals. In addition to being active in academic activities, he also works as an information technology consultant in local government, private and tourism. He can be contacted at email: msudarma@unud.ac.id.



Prof. Ida Ayu Dwi Giriantari    is a Professor in the Department of Electrical Engineering, Udayana University. She earned her bachelor's degree in Electrical Engineering from Udayana University, and continued her education at the University of New South Wales, Australia, for her master's and doctoral degrees in electrical power. She leads the Center for Community-Based Renewable Energy (CORE) at Udayana University and has expertise in renewable energy, energy management, energy tariffs, photovoltaics, smart grid, high voltage technology, and electrical circuits. She is also active in various international conferences and is a reviewer for several scientific journals. She can be contacted at email: dayu.giriantari@unud.ac.id.



Dewa Made Wiharta    holds an M.Eng. degree in Telecommunication Information System from Universitas Gadjah Mada in Jogjakarta, Indonesia, and is a lecturer at Udayana University in Denpasar, Bali, Indonesia. He earned his Ph.D. in Electrical Engineering from Institut Teknologi Sepuluh Nopember in Surabaya, Indonesia, in 2017. His research interests include multimedia signal processing and telecommunication engineering. He can be contacted at email: wiharta@unud.ac.id.