Security of Infocommunication Systems and Internet of Things

2025 Vol 3, No 1

https://doi.org/10.31861/sisiot2025.1.01002

Received 03 March 2025; revised 30 April 2025; accepted 07 June 2025; published 30 June 2025

IoT Based Smart Parking Systems in Unity With ML-agents Toolkit

Bohdan Dunaiev[1] and Oksana Olar[1,*]

[1]Departmet of Computer Systems and Networks, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine

*Corresponding author (E-mail: o.olar@chnu.edu.ua)

ABSTRACT Modern cities face problems with limited parking space, which requires effective optimization to reduce congestion. This paper presents an up-to-date approach to solving the problem of parking in restricted space conditions using the Unity Machine Learning Agents Toolkit (ML-Agents). This toolkit is based on artificial neural networks. It opens up wide opportunities for training agents capable of performing tasks in real time to optimize the use of parking areas and reduce the time to search for free spaces in real-time. Data transmission from sensors in Internet of Things (IoT) systems plays a key role in optimizing the use of parking spaces and increasing driver comfort. For this purpose, modern IoT technologies are integrated into the system, which allows the effective solving of parking problems in limited urban space conditions. The concept of an intelligent parking system is presented, which uses sensor technologies and machine learning algorithms to increase the efficiency of the process and driver comfort. Despite the high costs of installation and maintenance of traditional systems, the use of the Unity game engine and ML-Agents allows you to create training environments for preliminary testing, debugging, and optimization of algorithms. In the course of the research, a machine learning model was developed and analyzed using the Proximal Policy Optimization algorithm, which allowed us to reproduce various agent training scenarios. This contributed to the acceleration and stabilization of the training process, as well as the establishment of optimal model parameters based on the analysis of key performance metrics. The results of testing and comparative analysis confirm the prospects of the proposed approach in the field of autonomous car parking.

KEYWORDS Unity ML-Agents, sensors, tensor, reinforcement learning, Proximal Policy Optimization.

I. INTRODUCTION

The tasks of teaching a car to park in a parking lot using machine learning is relevant from the point of view of optimizing the use of parking zones in cities [1].

However, when creating such systems, difficulties arise, such as recognizing traffic lanes, keeping them, and detecting and avoiding various obstacles, which makes autonomous driving a difficult task. When parking a car, the driver must consider such factors as spatial awareness, trajectory planning real-time decision-making, etc.

The system uses sensor technology and a machine learning algorithm to provide efficient parking and convenience for drivers. The integration of this system with modern Internet of Things (IoT) technologies allows solving parking problems in cities where space is a limited resource [2].

Smart parking systems can also reduce traffic as drivers can find available parking spaces faster, reducing search time and improving overall city mobility. In addition, they can contribute to energy conservation by using automatic control of lighting and power supply in parking lots [3]. Developing an intelligent parking system is complicated by high installation and maintenance costs. Deployment of sensors is a rather expensive task [4]. However, it is possible to use the Unity game engine and the Unity Machine Learning Agents Toolkit (ML-Agents) plugin for initial training, testing, tuning, and creating a training environment.

A significant advantage of this method of system development is that all development takes place without any costs. At the same time, when developing the system, you can easily adjust the hyperparameters of the model and monitor how it will affect the smart parking system. Unity also provides the ability to easily create desired environments, such as underground parking or different types of surface parking. This allows you to easily test the developed model and improve it without any rental costs and various breakdowns.

The rest of the article is structured as follows. The next section examines a general approach to creating intelligent parking systems parking in limited spaces and describes the benefits of integrating a smart parking system into urban infrastructure. Section IV presents and explores the algorithms for training neural networks. Section V provides evaluations of different types of neural network architectures. Section VI concludes with findings and suggestions for future work.

II. A GENERAL APPROACH TO CREATING INTELLIGENT PARKING SYSTEMS

With the development of modern cities and automobiles, a large number of problems with traffic jams arise. Studies have shown that approximately 35% of cars in the city are looking for a parking space, which increases traffic congestion and causes traffic jams. The concept of using the Internet of Things for smart cities offers many solutions using different technologies.

The paper [5] describes the IoT-SPMS-LoRaWAN intelligent parking management system, which uses

Internet of Things sensors and Long Range Wide Area Network (LoRaWAN) technology to monitor parking space availability in real time. The authors show that the use of LoRaWAN provides wide network coverage and energy efficiency. In addition, the system operates independently of battery-powered solar panels. The implementation of IoT-SPMS-LoRaWAN in urban environments is expected to reduce traffic congestion, optimize parking space utilization, and increase driver awareness of available parking spaces. Research [6] describes a good use of the IoT framework. The authors introduce an intelligent parking system (IPS) built upon an IoT framework that gathers real-time data, transmits it to the cloud, and recommends suitable nearby parking spaces to users. An essential component of this framework is a mobile application, which allows users to view the availability of nearby parking spaces and conveniently reserve them. The system employs various IoT technologies, including Raspberry Pi, NodeMCU, radio frequency identification (RFID), and infrared (IR) sensors. Although these technologies may not offer optimal performance compared to IoT-SPMS-LoRaWAN solutions, they still demonstrate commendable results. Additionally, the incorporation of a dedicated mobile application significantly enhances the overall usability and effectiveness of the IPS.

The use of IoT technologies in the parking areas of smart cities shows good results, helps to reduce traffic jams, and improves the quality of life. Also, an important advantage of implementing smart parking is keeping the environment clean.

Research [7] cites many environmental benefits for modern cities, as smart parking allows to reduce traffic in the city by about 35%. Optimizing the allocation of public parking spaces is a key step in developing an intelligent parking system and further integrating it into the smart city infrastructure. Pape [8] proposes the use of a new mathematical model that describes the problem of optimal parking allocation, as well as an evolutionary algorithm to demonstrate how this model can be used in practice. The findings clearly indicate that the proposed approach effectively reduces overall parking costs and delivers immediate advantages to users.

In [9], a comprehensive study, comparison, and broad analysis of Smart Parking Systems (SPS) is provided in terms of technological approach, sensors used, network technologies, user interface, computing approaches, and services provided.

Based on a detailed review and analysis, it can be concluded that future smart cities will mostly use SPS with different approaches, where IoT will serve as the underlying technology. Users will interact with the system mainly through mobile applications, typically providing functions such as parking management, online payment, space reservation, and navigation to the car.

Sensor selection within an SPS will depend on multiple internal and external factors. Nevertheless, primary considerations when choosing sensors will include ease of installation, privacy protection, the accuracy of measurement methods, and the area covered by the sensors.

As a result, all the reviewed studies propose different SPS approaches and smart parking systems that can meet the high demand for smart parking. Different types of SPS have both advantages and disadvantages in solving various problems faced by systems. Also, according to the cited studies, the authors face such problems as the difficulty of setting the compatibility of various technologies, and the high cost of their testing. Most works do not consider the use of a virtual environment for the study of a SPS, testing, and final adjustment of the technologies used.

III. THE CONCEPT OF AN INTELLIGENT PARKING SYSTEM WITH ML-AGENTS

In this section, the ML-Agents Toolkit is applied, which is based on artificial neural networks and opens up wide opportunities for training agents to solve complex parking tasks in real-time. This can lead to the optimization of the use of parking zones and the reduction of time spent by drivers searching for free spaces.

This solution is very promising as, unlike various alternatives, it requires no financial investments and allows for the precise creation of a virtual model of a smart parking system.

The training environment includes two Unity components that help organize the Unity scene [10]:

– Agents attached to Unity GameObjects (e.g., any character in the scene) generate observations, perform the resulting actions, and provide rewards (positive or negative) as needed. Each agent is associated with a specific behavior.
– Behavior defines specific attributes of the Agent, such as the number of possible actions it can perform. Behavior can be considered a function that receives observations and rewards from the Agent and returns the corresponding actions.

An example of the ML-Agents Toolkit scheme based on an artificial neural network is shown in Fig. 1.

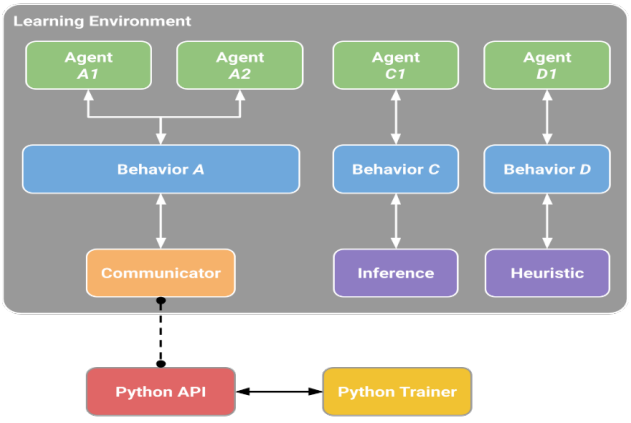

FIG. 1. ML-Agents Toolkit Diagram.

The model is trained using the Proximal Policy Optimization (PPO) algorithm, specifically the PPO-Clip variant. PPO is a first-order reinforcement learning algorithm that has emerged as a state-of-the-art method for policy optimization in on-policy settings. It builds upon the foundation of policy gradient methods and is conceptually derived from Trust Region Policy Optimization (TRPO), but with a significantly simpler

implementation and computational overhead. While TRPO explicitly enforces a constraint on the KL divergence between the new and old policies to ensure conservative updates, PPO-Clip instead incorporates a clipped surrogate objective function that implicitly limits the extent of policy change during training. This clipping mechanism restricts the probability ratio between the new and old policy within a fixed interval, thereby preventing excessively large updates that could destabilize learning.
Unlike methods that require explicit KL divergence penalties or complex second-order optimization, PPO-Clip achieves a trade-off between performance and simplicity. It enables efficient and stable learning by allowing multiple epochs of mini-batch updates per interaction with the environment, making better use of collected samples. Moreover, PPO demonstrates robust performance across a variety of continuous and discrete control tasks, while requiring minimal tuning of hyperparameters. This balance of stability, sample efficiency, and ease of implementation has contributed to PPO becoming one of the most widely adopted algorithms in reinforcement learning research and practical applications.

The Unity ML-Agents Toolkit also provides the ability to use various training scenarios, each offering its own advantages and having its own limitations or constraints. Simultaneous Single-Agent is a training process ideally suited for training an agent to park. It also helps accelerate and stabilize the training process.

The scheme of the algorithm is shown in Fig. 2.

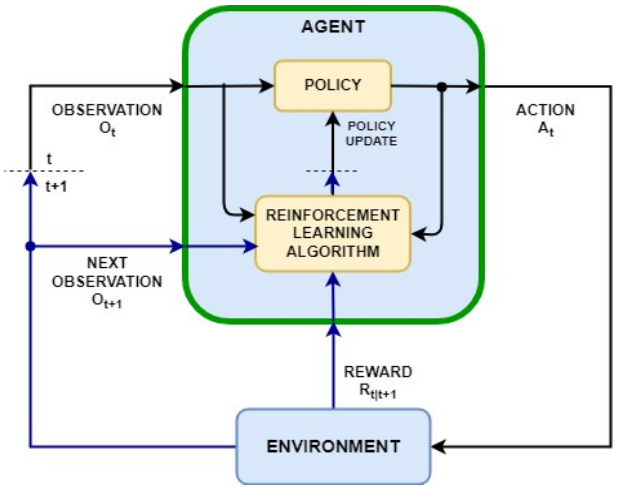

FIG. 2. Proximal Policy Optimization Algorithm Diagram.

This learning scenario can be described as training multiple independent agents, each receiving a separate reward signal but interacting through a common Brain component. Additionally, the ML-Agents Toolkit provides the ability to use a TensorFlow utility called TensorBoard [11]. This utility allows you to set custom metrics for evaluating training, in addition to those provided by the framework. This ensures training effectiveness and helps determine the optimal model parameters to achieve the desired results.

Therefore, the integration of machine learning into parking systems can significantly contribute to solving the problems of overall parking space scarcity and disorganization in this aspect.

IV. DESCRIPTION OF THE TRAINING ALGORITHM

PPO [12] is motivated by the same question as TRPO, namely how to make the largest possible step in policy improvement using available data without going so far as to accidentally cause a drop-in productivity?

While TRPO addresses this issue using a complex second-order optimization approach, PPO encompasses a set of simpler first-order methods that rely on alternative techniques to maintain new policies close to previous ones. PPO methods are comparatively straightforward to implement and have empirically shown performance at least equivalent to TRPO.

PPO includes two primary variants:

1. PPO-Penalty;
2. PPO-Clip.

PPO-Penalty addresses KL-constrained policy updates similarly to TRPO but introduces a penalty for KL divergence within the objective function, rather than enforcing it as a strict constraint. Additionally, it automatically adjusts the penalty coefficient throughout training. In contrast, PPO-Clip does not include a KL-divergence component or any explicit constraints in its objective. Instead, it employs a specific clipping mechanism within the objective function to discourage significant deviations of the new policy from the previous policy.

This discussion will exclusively concentrate on PPO-Clip, which is the main variant employed by OpenAI.

PPO-clip policy update through

$$\theta_{k+1} = \arg\max_{\theta} \underset{s,a\sim\pi_{\theta_k}}{E} [L(s,a,\theta_k,\theta)], \tag{1}$$

typically using multiple stages (usually mini-series) of SGD to achieve the maximum goal.

Where in the formula 0 is the initial policy parameter and is the policy in the given environment. Here, L is given

$$L(s,a,\theta_k,\theta) = \min(\frac{\pi_\theta(a\,|\,s)}{\pi_{\theta_k}(a\,|\,s)} A^{\pi_{\theta_k}}(s,a),$$
$$clip(\frac{\pi_\theta(a\,|\,s)}{\pi_{\theta_k}(a\,|\,s)}, 1-\epsilon, 1+\epsilon) A^{\pi_{\theta_k}}(s,a)). \tag{2}$$

This expression contains ϵ is a (small) hyperparameter that approximately controls how much the new policy is allowed to differ from the old one, with A representing the advantage estimate.

At first glance, the formula appears complicated, making it challenging to immediately understand its function or how it maintains proximity between new and old policies.

However, a significantly simplified version of this objective exists, which is easier to interpret and implement

$$L(s,a,\theta_k,\theta) = \min(\frac{\pi_\theta(a\,|\,s)}{\pi_{\theta_k}(a\,|\,s)} A^{\pi_{\theta_k}}(s,a), g(\epsilon, A^{\pi_{\theta_k}}(s,a))), \tag{3}$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)\,A & A \geq 0 \\ (1-\epsilon)\,A & A < 0. \end{cases} \tag{4}$$

To understand the intuition behind this, let's consider state-action pairs individually and examine two scenarios: one with a positive advantage and another with an opposing advantage.

When the advantage is positive, meaning the action should ideally become more probable, the contribution to the objective is expressed as

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a\mid s)}{\pi_{\theta_k}(a\mid s)},(1+\epsilon)\right)A^{\pi_{\theta_k}}(s,a). \quad (5)$$

Given a positive advantage, increasing the likelihood of selecting the action will enhance the objective. However, the presence of the minimum function imposes a limit, preventing excessive deviation from the previous policy.

Once $\pi_\theta(a\mid s) > (1+\epsilon)\pi_\theta(a\mid s)$, the minimum term activates, reaching its upper limit, $(1+\epsilon)A^{\pi_{\theta_k}}(s,a)$. Thus, significant deviations from the old policy provide no further benefit.

When the advantage is negative, meaning the state-action pair is unfavorable, the expression simplifies to

$$L(s,a,\theta_k,\theta) = \max\left(\frac{\pi_\theta(a\mid s)}{\pi_{\theta_k}(a\mid s)},(1-\epsilon)\right)A^{\pi_{\theta_k}}(s,a). \quad (6)$$

Since the preference is negative, the objective improves when the action's likelihood decreases, meaning when $\pi_\theta(a\mid s)$ is reduced. However, the maximum value in this term restricts how much the target can actually increase.

Once $\pi_\theta(a\mid s) < (1-\epsilon)\pi_{\theta_k}(a\mid s)$, the expression reaches its peak value, specifically $(1-\epsilon)A^{\pi_{\theta_k}}(s,a)$. Thus, significantly deviating from the previous policy does not yield additional benefits.

Therefore, the cutoff acts as a regularizer by discouraging significant policy changes. The hyperparameter epsilon effectively sets the allowable deviation of the new policy from the old one while still enabling the objective to be optimized.

V. EVALUATING MODEL TRAINING

The following values were determined as the main metrics of educational success (i.e., effectiveness of training):

- The probability of successful parking;
- Minimization of the spatial and angular deviation between the center of the vehicle and the center of the designated parking space;
- Reduction in the number of collisions between the agent and environmental elements, including curbs and other vehicles.

To monitor these indicators, custom data logging mechanisms were implemented directly within the agent's software code, enabling real-time collection of performance-relevant data during training episodes.

In the context of reinforcement learning, effectiveness is understood as the degree to which an agent improves its behavior to achieve task-specific objectives in a stable, sample-efficient, and generalizable manner. From a quantitative perspective, effectiveness was assessed using the aforementioned metrics, which reflect both the success rate (e.g., probability of correct parking) and the precision of task execution (e.g., alignment accuracy, collision avoidance). These metrics were tracked across training steps to evaluate learning progression, policy convergence, and overall performance consistency.

In addition to quantitative measurements, qualitative assessment was conducted through direct visual inspection of the agent's behavior within simulation environments. These evaluations included trajectory smoothness, decision-making rationality (e.g., avoidance of unnecessary steering), and robustness to changes in initial conditions or minor perturbations in the environment. This dual-pronged assessment strategy – combining objective metrics with subjective behavior evaluation – provided a comprehensive perspective on the effectiveness of the learning process.

Unity ML-Agents offers several training paradigms, each with its own advantages and limitations. During experimentation, the parking task was initially trained using a Single-Agent scenario, in which only one agent interacts with the environment at a time. However, it was later determined that the Simultaneous Single-Agent configuration – in which multiple agents are trained independently in parallel, each receiving its own reward signal but sharing a common policy (Brain component) – yielded superior results.

This parallelized training configuration enabled significantly faster data collection per episode, leading to improved sample efficiency and more stable learning dynamics. Furthermore, the use of independent reward signals for each agent preserved the diversity of learning experiences while ensuring convergence toward a common optimal behavior. This setup was especially advantageous for tasks involving spatial complexity and fine motor coordination, such as autonomous parking.

To compare the two training paradigms, experiments were conducted on two separate training environments. One employed the Single-Agent configuration, while the other used Simultaneous Single-Agent training. Figure 3

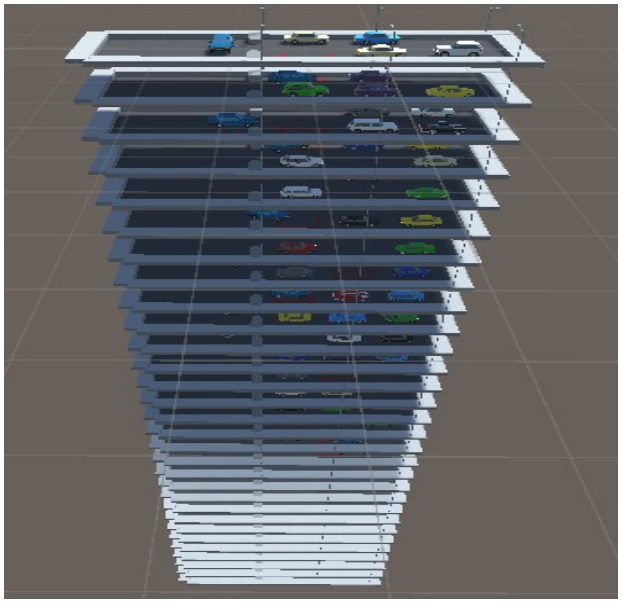

FIG. 3. Example of a scene with 30 parking lots and agents training on them.

illustrates a representative training scene containing 30 parking spaces, each occupied by an agent learning independently within a shared policy framework.

After completing 2.5 million training steps, performance statistics were collected for both configurations based on the established evaluation metrics. For clarity, results corresponding to the Simultaneous Single-Agent scenario are displayed in light blue, while those from the Single-Agent configuration are shown in dark blue across the following figures.

In Fig. 3 shows an example of a scene with 30 parking lots and agents trained on them.

Figure 4 presents the angular deviation between the agent's heading and the target parking direction. The Simultaneous Single-Agent configuration showed greater stability and faster convergence in minimizing this angle, suggesting more effective orientation control during parking.

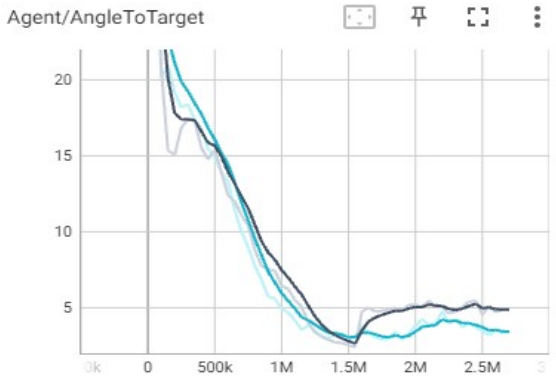

FIG. 4. The value of the angle between the center of the vehicle agent and the center of the parking space.

Figure 5 illustrates the reduction in distance between the agent's center and the center of the parking space over time. The results clearly indicate higher spatial accuracy in the Simultaneous Single-Agent setup.

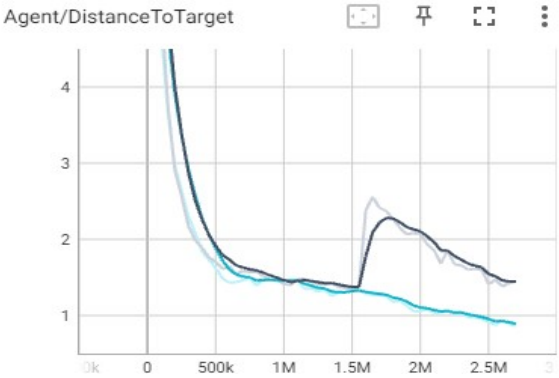

FIG. 5. The value of the distance between the center of the car agent and the center of the parking space.

Figure 6 tracks the number of collisions with curbs across training episodes. Again, the Simultaneous Single-Agent scenario consistently outperformed the Single-Agent baseline by demonstrating fewer collisions, reflecting safer and more controlled behavior.

Based on these results, it can be concluded that the Simultaneous Single-Agent training configuration is more effective in terms of both learning speed and quality of the learned policy. The trained model is capable of executing accurate and robust parking maneuvers from a wide range of starting conditions and can therefore be considered a strong candidate for future deployment in intelligent parking systems. These findings support further investigation into the model's scalability and its integration into real-world autonomous parking infrastructures.

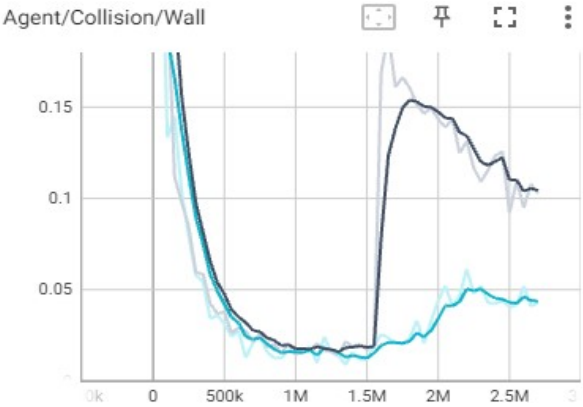

FIG. 6. The number of collisions of the agent with the curb.

V. CONCLUSION

Through the integration of the Unity ML-Agents framework, it has been demonstrated that virtual agents can be trained to autonomously navigate and park vehicles in various scenarios that accurately reproduce real-world parking conditions. The ML-Agents-based approach, especially in the context of Unity's powerful and versatile simulation, has proven to be an effective method for developing adaptive and intelligent parking systems. The next step in the research is to extend the model to train the agent to park in different types of parking spaces and extend the smart parking system by adding user interaction with the city infrastructure and the parking itself.

In addition, we plan to integrate real-time traffic data to improve the adaptability of the system to dynamic urban environments and explore advanced reinforcement learning techniques to further optimize agent performance. These improvements are aimed at increasing the efficiency and effectiveness of the system in real-world applications, ultimately contributing to smarter and more responsive solutions for urban mobility.

AUTHOR CONTRIBUTIONS

B.D. – conceptualization, software, resources, writing of the original, preparation of the draft, visualization, testing, results; O.O. – methodology, research, preparation of the original, resources, writing-review and editing, supervision, verification, validation.

COMPETING INTERESTS

The authors declare no competing interests.

REFERENCES

[1] B. Padmavathi and V. Selvaraj, "Advanced optimization-based weighted features for ensemble deep learning smart occupancy detection network for road traffic parking," *J. Netw. Comput. Appl.*, vol. 230, p. 103924, 2024. doi: 10.1016/j.jnca.2024.103924.

[2] F. Caicedo, "Real-time parking information management to reduce search time, vehicle displacement and emissions," *Transp. Res. D: Transp. Environ.*, vol. 15, no. 4, pp. 228–234, 2010. doi: 10.1016/j.trd.2010.02.008.

[3] S. Saki and T. Hagen, “Cruising for parking again: Measuring the ground truth and using survival analysis to reveal the determinants of the duration,” *Transp. Res. A: Policy Pract.*, vol. 183, p. 104045, May 2024. doi: 10.1016/j.tra.2024.104045.
[4] W. A. Jabbar, L. Y. Tiew, and N. Y. Ali Shah, “Internet of things enabled parking management system using long range wide area network for smart city,” *Internet Things Cyber-Phys. Syst.*, vol. 4, pp. 82–98, 2024. doi: 10.1016/j.iotcps.2023.09.001.
[5] J. Arellano-Verdejo, F. Alonso-Pecina, E. Alba, and A. Guzmán Arenas, “Optimal allocation of public parking spots in a smart city: problem characterization and first algorithms,” *J. Exp. Theor. Artif. Intell.*, vol. 31, no. 4, pp. 575–597, 2019. doi: 10.1080/0952813X.2019.1591522.
[6] R. Ke, Y. Zhuang, Z. Pu, and Y. Wang, “A smart, efficient, and reliable parking surveillance system with edge artificial intelligence on IoT devices,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 4962–4974, 2021. doi: 10.1109/TITS.2020.2984197.
[7] A. Aditya, Sh. Anwarul, R. Tanwar, and S. Krishna Vamsi Koneru, “An IoT assisted Intelligent Parking System (IPS) for Smart Cities,” *Procedia Comput. Sci.*, vol. 218, pp. 1045–1054, 2023. doi: 10.1016/j.procs.2023.01.084.
[8] M. Choi, G. Kang, and S. Lee, “Autonomous driving parking robot systems for urban environmental benefit evaluation,” *J. Cleaner Prod.*, vol. 469, p. 143215, 2024. doi: 10.1016/j.jclepro.2024.143215.
[9] A. Fahim, M. Hasan, and M. Alam Chowdhury, “Smart parking systems: comprehensive review based on various aspects,” *Heliyon*, vol. 7, no. 5, p. e07050, May 2021. doi: 10.1016/j.heliyon.2021.e07050.
[10] T. Liu, H. Chen, J. Hu, Z. Yang, B. Yu, X. Du, Y. Miao, and Y. Chang, “Generalized multi-agent competitive reinforcement learning with differential augmentation,” *Expert Syst. Appl.*, vol. 238, part C, p. 121760, 2024. doi: 10.1016/j.eswa.2023.121760.
[11] M. Matulis and C. Harvey, “A robot arm digital twin utilizing reinforcement learning,” *Comput. Graph.*, vol. 95, pp. 106–114, 2021. doi: 10.1016/j.cag.2021.01.011.
[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017. doi: 10.48550/arXiv.1707.06347.

Bohdan Dunaiev

In 2024, graduated from Yuriy Fedkovych Chernivtsi National University with a degree in Computer Engineering, specialized in “Computer Engineering of Internet of Things Technologies and Cyber-Physical Systems” (Master's level).

ORCID ID: 0009-0000-4131-8590

Oksana Olar

In 2010, defended her PhD thesis in the specialty "Computer Systems and Components". Currently, works as an associate professor at the Department of Computer Systems and Networks of Chernivtsi National University. Field of scientific interests: data processing and analysis, machine learning, IoT tools.

ORCID ID: 0000-0001-8574-1980

Система розумного паркування на основі IoT з використанням Ml-Agents Toolkit

Богдан Дунаєв[1], Оксана Олар[1,*]

[1] Кафедра комп’ютерних систем та мереж, Чернівецький національний університет імені Юрія Федьковича, Чернівці, Україна

*Автор-кореспондент (Електронна адреса: o.olar@chnu.edu.ua)

АНОТАЦІЯ У сучасних містах виникають проблеми із обмеженим паркувальним простором, який потребує ефективної оптимізації для зменшення заторів. У цій роботі представлено актуальний підхід до вирішення проблеми паркування в умовах обмеженого простору з використанням Unity Machine Learning Agents Toolkit (ML-Agents). Даний інструментарій базується на штучних нейронних мережах та відкриває широкі можливості для навчання агентів, здатних у реальному часі виконувати завдання щодо оптимізації використання паркувальних зон та скорочення часу пошуку вільних місць у реальному часі. Передача даних із датчиків у системах Інтернету речей (IoT) відіграє ключову роль в оптимізації використання паркувальних місць і підвищенні комфорту водіїв. Для цього в систему інтегровано сучасні технології IoT, які дозволяють ефективно вирішувати проблеми паркування в умовах обмеженого міського простору. Представлено концепцію інтелектуальної системи паркування, яка використовує сенсорні технології та алгоритми машинного навчання для підвищення ефективності процесу та комфорту водія. Незважаючи на високі витрати на монтаж та обслуговування традиційних систем, застосування ігрового рушія Unity та ML-Agents дає змогу створювати навчальні середовища для попереднього тестування, налагодження та оптимізації алгоритмів. У процесі дослідження було розроблено та проаналізовано модель машинного навчання з використанням алгоритму Proximal Policy Optimization, що дозволило відтворити різноманітні сценарії навчання агентів. Це сприяло прискоренню та стабілізації процесу навчання, а також встановленню оптимальних параметрів моделі на основі аналізу ключових метрик ефективності. Отримані результати тестування та порівняльного аналізу підтверджують перспективність запропонованого підходу у сфері автономного паркування автомобілів.

КЛЮЧОВІ СЛОВА Unity ML-агенти, датчики, тензор, навчання з підкріпленням, оптимізація проксимальної політики.