Computer Science Department

Final Report Document for Senior Project – Spring 2024

Title of the project:	"Parking Assistance Application using IoT Devices and Machine Learning on the Edge"
Team Members:	Aruzhan Amangeldina, Ruana Saduakhas, Batyrkhan Baimukhanov, Zhanabek Tuleshov
Project Advisor/Co-Advisors	Assistant Professor Dimitrios Zormpas

1. Executive Summary

During the Senior Project, we investigated the deployment of compact convolutional neural networks (CNNs) on edge devices for parking slot occupancy classification, addressing the growing issue of inefficient urban parking. Two CNN architectures were selected for evaluation from other research, namely mAlexNet and a model proposed by Mohan et al. [9]. In addition, a smaller version of mAlexNet was devised during the project to address the strict memory constraints of the edge devices used. The three models were trained using uniform training parameters on a publicly available parking dataset. After applying full-integer post-training quantization, the models were deployed and tested on OpenMV Cam H7 and Raspberry Pi Model B edge devices. Model performances were evaluated using a hand-made parking lot replica, in which distinct camera angles, day-night cycles, as well as various weather conditions such as snow and rain were simulated. While mAlexNet on Raspberry Pi achieved the best accuracy among the tested models, other lightweight models running on OpenMV also delivered satisfactory results, exceeding 90% in accuracy. Based on the results of the study, our version of mAlexNet and Mohan's model on OpenMV's camera appear well-suited for real-world deployment and the affordability of OpenMV devices further strengthens this case.

2. Introduction

The increasing number of cars in urban environments creates the necessity for efficient parking solutions. With the expanding urban areas, the issue of parking scarcity is being intensified, thus extending the time spent by drivers to find vacant spaces. This phenomenon leads to increased fuel consumption, traffic congestion, and emissions [1]. It was documented that the quest for parking can account for as much as 30% of traffic congestion in urban centers [2], with an average driver wasting approximately 7.8 minutes on this task [3]. Traditional methods of parking management typically involve either sensor-based systems or manual monitoring of vehicle movements, such as counting the number of entering and exiting vehicles. These methods proved to be costly, labor-intensive, and prone to inaccuracies [2].

Our proposed solution encompasses the development of a real-time parking monitoring and management system that utilizes battery-powered resource-constrained edge devices equipped with cameras. The system incorporates a convolutional neural network (CNN) executed directly on the device, ensuring enhanced operational capabilities and responsiveness due to computing on the edge, bypassing network delays that are present in the cloud-computing approach. By determining the availability of free parking slots prior to a driver's entry into a parking lot, the system aims to decrease the time and energy spent by drivers in search of parking. This enhancement in convenience aligns with broader smart city initiatives and underscores the utility of edge-based machine learning in providing cost-effective, adaptable, and accurate parking solutions.

A number of approaches have been proposed recently to identify vacant parking slots with the help of Machine Learning (ML) [2][7][8]. These approaches can be used to automatically inform drivers about available parking space, and thus, mitigate the issues of congestion and pollution in large cities. This project provides an evaluation of two known ML approaches for parking slot detection, the mAlexNet [8] and Mohan's models [9], and assesses the feasibility of deploying them on resource-constrained Internet of Things devices. Since these devices may run on batteries, energy consumption is an important feature that has to be considered during their deployment. Moreover, constraints in terms of computation power, available memory, and storage usually apply. A series of experiments are conducted to assess the performance of the two models on a Raspberry Pi 4 Model B as well as on a much less capable IoT device, OpenMV Cam H7. Due to the large size of the first model, a reduced version was devised and tested on the OpenMV device.

The report is organized as follows. Section 3 delves into the Background/Related Work, providing a comprehensive review of existing literature on parking occupancy detection and available lightweight models. Section 4 details the Project Approach, extensively describing the technical approach, data acquisition, machine learning models used, and the overall system architecture for our parking assistance solution. In Section 5, we present the Execution of our system, where the effectiveness of the proposed solution is assessed through various tests and simulations, demonstrating its accuracy and reliability in different scenarios. Finally, Section V offers the Conclusion, summarizing the key findings, discussing the implications of this study for smart city initiatives, and suggesting directions for future research in this area.

3. Background/Related Work

A review of prior work on parking surveillance and similar tasks was conducted to analyze available lightweight machine-learning models that can be deployed on embedded systems. The emphasis was on the image classification models to differentiate between available and busy parking slots presented with an input in the form of a parking space patch.

The early works on parking surveillance using camera feeds were conducted by [4], [5], and [6]. However, these works primarily used rule-based and image-processing techniques and faced challenges in scalability and generalization.

Later works adopted a strategy of taking widely recognized deep learning architectures as a basis and reducing their complexity in order to operate on edge devices. Such an approach was taken by Amato et al. in [7] and [8]. In [7], the authors introduced the application of deep learning techniques to enhance

the system's generalization abilities to diverse parking and lighting conditions. In [8], the authors proposed a decentralized solution for smart cameras using a deep convolutional neural network to detect parking occupancy. Namely, the authors propose a minimized version of the AlexNet model, termed mini AlexNet (mAlexNet), 1340 times smaller in number of parameters than the original model. The size reduction is justified by the simplified nature of the binary classification task at hand. The authors utilized Raspberry Pi 2 model B equipped with a camera module and achieved comparable performance with respect to the original model, scoring over 95% in accuracy across various experiments.

A similar methodology was employed by Ellis et al. [2] to detect parking availability. Building upon the concept of mAlexNet, the authors introduced Mini DenseNet and Simple DenseNet architectures, simplified versions of the DenseNet model. Despite the multifold reduction in size, the resulting models achieved an average accuracy of over 99%, proving the diminishing increase in accuracy as the model size grows. The architectures were tailored for the deployment of CCTV devices. However, due to closer alignment in hardware setup with [8], it was decided to follow mAlexNet architecture for our work.

Mohan et al. [9] introduced a novel architecture to detect face mask usage in public settings with low-memory OpenMV Cam H7. This work applies to our case as the authors address a similar binary classification problem (mask and no mask) and utilize the same edge device. The authors applied a low-bit quantization method for model size reduction by transforming weight values from floats to integers. This approach along with the TensorFlow Lite framework used in the paper was employed for our case to adapt to low-resource edge devices. The proposed architecture achieved an accuracy of over 99% in the face mask detection task.

4. Project Approach

The proposed solution is a parking lot occupancy detection system that consists of camera-equipped edge devices installed in a parking lot that run convolutional neural network (CNN) models. The primary workflow of the proposed system is outlined as follows:

1. A camera-equipped edge device with a pretrained CNN model is installed in a parking lot.
2. Coordinates of each parking slot are manually input into the device.
3. The edge device periodically captures images, which are then divided into patches representing individual parking slots according to the predefined coordinates.
4. The patches are fed to the CNN model sequentially.
5. Depending on the inference results from the model, the parking slot is colored with green if it is vacant and with red if occupied.
6. The user searching for a vacant parking spot sends a request to a Telegram bot.
7. The Telegram bot accesses the most recently saved image and sends it to the user.
8. The user then proceeds to the chosen parking spot and parks their vehicle there.
9. The new vehicle in the parking lot is detected by the system and the corresponding parking slot is updated as occupied.

The latter part of this section is dedicated to provide an overview of the hardware setup and the details of model training methodology, including a description of a dataset, model architectures, training specifications, and quantization. Finally, third-party tools utilized during the project are listed in the end.

Hardware Setup

During the project we employed OpenMV Cam H7 and Raspberry Pi 4 Model B as edge devices in the proposed smart parking system.

1) *OpenMV Cam H7:* OpenMV Cam H7 is a low-power, low-memory, small microcontroller unit equipped with a 32-bit ARM Cortex-M7 processor and OV7725 image sensor capturing images in RGB565 format. Operating on the MicroPython operating system, it provides a means for developing and deploying image processing and machine learning applications on the edge [10]. The OpenMV Cam H7 faces resource management challenges, with a limited frame buffer RAM of 496KB. Consequently, we aimed for a model size of under 200KB to ensure that the combined memory footprint of image capture and processing, model loading, tensor allocation, and the microcontroller's internal operations remained within acceptable limits. OpenMV Cam H7 is illustrated in Figures 1-2.

2) *Raspberry Pi:* The Raspberry Pi 4 Model B device, coupled with the Raspberry Pi Camera Board v1.3 featuring a 5-megapixel sensor, provided an alternative platform for our autonomous parking assistance system. In contrast to the OpenMV Cam H7, the Raspberry Pi offered greater flexibility in terms of computational resources and expandability. We utilized the Raspberry Pi for testing models that were not compatible with the OpenMV Cam H7 due to memory constraints. Raspberry Pi Model 4 B and Pi Camera are demonstrated in Figures 3-4.

Data Description

The primary dataset utilized for this project was the CNRPark+EXT dataset, an extension of the CNRPark dataset, collected by Amato et al. [8]. CNRPark+EXT contains around 150,000 150x150 px images across 164 parking lots collected by 9 cameras from November 2015 to February 2016 [8]. The dataset offers various lighting and weather conditions, including sunny, rainy, and overcast weather scenarios, see Figure 5. The images depict cropped patches of individual parking slots within parking lots. The dataset classifies the parking slots into 2 classes: occupied and unoccupied. The division between the 2 classes is not perfectly balanced with the occupied class representing 54.71% and the unoccupied class comprising 45.29% of the dataset. Multiple splits, categorized by weather conditions and cameras, are available, with the training (65%), validation (13%), and test (22%) splits being employed for this project. The CNRPark+EXT dataset is available at [11].

Model Architectures

Our investigation primarily centered on two CNN model architectures: mAlexNet and a model proposed by Mohan et al. [9]. We tailored these architectures by adjusting their final fully connected layers to consist of two output neurons with sigmoid activation functions, each representing probabilities for occupancy and non-occupancy. Mini AlexNet (mAlexNet) architecture was proposed in [8] as a mini version of the AlexNet to address a binary classification of parking slots. The model accepts images of size 224 by 224 pixels and has 3 convolutional layers with 32380 trainable parameters. mAlexNet architecture is illustrated in Fig. 6. Despite its small size of 39KB in quantized form, we were unable to test this model with the OpenMV device due to high memory requirements for the input layer. The model was tested with a Raspberry Pi device instead. During this study, a smaller version of mAlexNet was implemented. By reducing the input size to 32 by 32 pixels as well as consequent kernels, we were able to decrease the number of model parameters to almost one-third of the original count (12020 trainable parameters) and fit it into the OpenMV device. A fully quantized version of the model has a file

size of 19KB. In this work, the smaller variant is referred to as mAlexNet32, while the original mAlexNet is referred to as mAlexNet224 to reflect the input size of an image (Fig. 7). A slightly larger CNN model with dropout layers proposed in [9] was also tested. Originally designed for medical face mask detection, we repurposed the architecture for the parking slot classification scenario. This model comprises three convolutional layers and 128,226 trainable parameters. The quantized version's file size is 137 KB, making it suitable for testing on the OpenMV microcontroller. The model is referred to as Mohan's model (Fig. 8)

Training Specifications

For the comparative analysis of the above-mentioned architectures, a consistent set of training specifications was implemented to ensure fair evaluation. The chosen loss function for all models was Categorical Cross Entropy. To facilitate efficient optimization, the Adam optimizer was uniformly applied. A fixed learning rate of 0.001 was maintained throughout the training process. In order to mitigate overfitting and enhance training efficiency, early stopping with a patience parameter set to 3 was applied. This mechanism halted training if there was no improvement in validation performance for three consecutive epochs, preventing unnecessary computation and potential overfitting. These standardized training specifications ensured consistent comparative analysis of the models.

Post-training Quantization

In this phase, we applied post-training full integer quantization, converting the trained models from float-32 precision to int-8 precision using TensorFlow Lite's Full Integer Quantization technique. This method involved utilizing a representative dataset, extracted as a small subset of the validation dataset. Weight quantization was crucial for several reasons:

1) *Model size reduction:* Quantization allows for a significant reduction in model size, approximately by a factor of four. This reduction in size is particularly advantageous for resource-constrained devices that typically have limited memory and storage capacities.

2) *Compatibility with microcontroller units:* Since most microcontroller units lack support for floating-point arithmetic, quantization becomes an essential step to ensure compatibility with the hardware constraints.

3) *Improved computational efficiency:* By employing lighter models that rely solely on integer arithmetic, computational requirements for edge devices are reduced. Consequently, this leads to faster inference times and energy savings due to quicker execution. These improvements are especially valuable in real-time applications where low latency is critical.

Third-party tools

1. OpenMV IDE was utilized to develop and deploy scripts into the OpenMV microcontroller.
2. The Python-telegram-bot library facilitated the development process of the Telegram bot.
3. The RPC (Remote Procedure Control) library enabled the control and execution of commands on a camera from a PC.
4. TensorFlow Lite was used to deploy machine learning models on resource-constrained platforms.

5. Project Execution

Fall semester:

A thorough literature review was conducted to identify relevant existing information pertaining to our research topic. This review led us to discover two main architectures: mAlexNet and Mohan's model. When deploying these architectures in OpenMV, we encountered a challenge with mAlexNet due to the microcontroller's memory restrictions. OpenMV Cam H7 has a limited available frame buffer RAM of 496 KB, which is used to store both an image and a model. Consequently, there was a trade-off in the allocation of frame buffer RAM between the image and the model, as increasing one reduces the available space for the other. For that reason, we aimed for a model size of under 200KB to ensure that the combined memory footprint of image capture and processing, model loading, tensor allocation, and the microcontroller's internal operations remained within acceptable limits. We found balance by setting QVGA resolution for the camera (320x240 pixels and 16-bit RGB results in 150 KB images) and modifying mAlexNet's architecture to match the input size of Mohan's model (32x32) and adjusting subsequent layers.

The models were trained using the CNRPark-EXT dataset as well as our own generated toy dataset and quantized to reduce their sizes. They were then deployed on the OpenMV device in a parking toy environment. To imitate real-world usage scenarios, a Telegram bot was created as a user interface for the users and connected to OpenMV so that the users could access the current data and decide. We made the classification results accessible to users through a Telegram bot connected to the device. However, the models' performance was not consistent and did not produce sufficient results.

Spring semester:

The main focus of the Spring semester was on the improvement of the model performance through the model optimization and adaptation of the existing and new solutions to the Raspberry Pi device. Consequently, the comparison of various model performances on the two devices under consideration was conducted.

Firstly, the Raspberry Pi equipped with a camera module was configured to run an image classification task. The code was written using cv2 and TensorFlow Lite libraries to take image tensor input of an individual parking slot at a predefined coordinate and feed it into the image classification models developed earlier. The original mAlexNet224 was tested on Raspberry Pi in order to compare its performance with its altered version mAlexNet32 run on an OpenMV device.

Next, efforts were made to scrutinize the model architecture and training specifications adopted earlier in the Fall semester in order to improve the performance of Mohan's and mAlexNet32 models. The issue was that the evaluation of models on the test subset of CNRPark-EXT rendered adequate results of over 90%, however, during the deployment of the same model on OpenMV and manual testing on real-time parking monitoring, the models did not produce consistent and sufficient results. The issue was addressed by identifying that the earlier models were not fully quantized. Input and output were in float32 format, which is incompatible with the OpenMV requirements. The second discovery was that Mohan's model we utilized was not totally identical to the model proposed in the paper [9]. Particularly, the fully connected layers at the end needed alterations. After the modifications in the model architectures, models' retraining, and full-integer quantization, the models produced improved results.

Another modification we introduced was training the models fully on the CNRPark-EXT dataset without the custom toy dataset. The extensive coverage of various parking conditions by the CNRPark-EXT may have improved the inference results.

Afterward, Mohan's model was evaluated on both OpenMV and Raspberry Pi devices. For Raspberry Pi, the size of the image tensor was resized to 32x32 in order to align with the expected input size.

Subsequently, the performance of parking occupancy detection of two devices was evaluated and compared. The modified mAlexNet32 and Mohan's models were run on an OpenMV device, while the original mAlexNet224 and Mohan's models were tested on Raspberry Pi. Corresponding evaluation procedures were carried out for each case and the details are further described in the Evaluation section. The developed system was also evaluated in a real-world setting by placing an OpenMV device with a deployed model to monitor a real parking environment through a window.

6. Evaluation

This section discusses the evaluation metrics of the given models' performance, namely mAlexNet224 & Mohan's on Raspberry Pi and mAlexNet32 & Mohan's model on OpenMV Cam H7, and then showcases the performance results and provides detailed elaboration on them.

Evaluation Methodology: This project applies a simulation-based evaluation approach to assess the models' performance by constructing synthetic environments to mimic real-world scenarios under controlled conditions. In our case, using toy cars to represent vehicles in a parking lot allows for the creation of diverse scenarios and controlled variables for testing. While these artificial environments may not perfectly replicate real-world conditions, they provide a valuable means of evaluating model performance in a controlled setting before deploying them in practical applications. Controlled variables include camera angles, light intensity, weather conditions, and obstacles.

1) *Different Camera Angles*: The perspective captured by a camera significantly influences the resulting image, underscoring the importance of considering multiple angles in performance evaluations. By examining models' performance across diverse viewpoints initially, we ensure their effectiveness remains constant regardless of angle. Furthermore, assessing their performance from various camera positions mirrors real-world deployment scenarios, enabling us to anticipate and address potential challenges that may arise due to differing perspectives. Hence, two camera angles were considered in the evaluation metrics for each model. The perception of the parking lot from 2 different camera angles at the Raspberry Pi device is illustrated in Figure 17.

2) *Light Intensity*: Light intensity plays a crucial role in determining the quality of an image, as it directly impacts the visibility and clarity of objects captured. Consequently, variations in light intensity can significantly affect the performance of models, potentially leading to a deterioration in their accuracy and reliability. Therefore, it is essential to evaluate the performance of these models under different lighting conditions. Thus this evaluation considers both normal lighting and reduced light conditions. The intensity of normal lighting during the experiments was 100 lux which is comparable to room lighting [12] and the intensity of reduced lighting ranged from 5 to 50 lux depending on the

distance from the light source. For comparison, the light intensity for street lighting is 15 lux [13]. The simulation of the light intensity on the toy parking setting is demonstrated in Figure 15.

3) *Weather conditions*: In our experiment, we endeavored to incorporate various weather conditions, specifically snow coverage and puddles formed after rain. This deliberate inclusion was motivated by the necessity to ensure the models' robustness across diverse environmental settings. Recognizing that these models might be deployed in geographical locations with varying weather conditions, it was imperative to assess their performance under such circumstances comprehensively. By simulating scenarios involving snow and rain-induced puddles, we aimed to gauge the models' adaptability and efficacy in handling adverse weather conditions. This holistic approach not only enhances the models' reliability but also underscores their suitability for deployment in real-world scenarios characterized by unpredictable weather patterns. The simulation of snow coverage and rain paddles are depicted in Figure 16.

4) *Obstacles*: In addition to toy cars, we employed various other toys to simulate removable objects within parking slots, accurately reflecting scenarios encountered in real-life environments. These objects were strategically chosen to represent potential obstacles such as people, cats, dogs, or any other items typically captured by cameras but can be removed when a vehicle approaches. Maintaining proportional accuracy was paramount to effectively replicate real-world scenarios. By integrating these removable objects into our simulated parking lots, we aimed to create realistic and dynamic environments that closely mirrored the challenges faced in actual parking scenarios. This comprehensive approach ensured that our evaluation encompassed a wide range of potential obstacles, thereby providing a thorough assessment of the models' performance in practical settings. Figure 18 illustrates the simulation of obstacles in a parking lot.

Results: The performance evaluation of our models was conducted, and the findings are presented in Tables 1-3.

Table 1 presents the size and accuracies averaged across all evaluation settings for each model and device combination. The results of the mAlexNet224 model are available for Raspberry Pi only due to compatibility with the required input size. The best accuracy was attained by the mAlexnet224 run on Raspberry Pi, which can be attributed to its larger input size, resulting in higher resolution and hence better inference abilities. The execution of mAlexNet32 on OpenMV produced slightly lower but still comparable performance. The evaluation of Mohan's model on both devices rendered compelling results with OpenMV producing slightly higher accuracy, which can be attributed to better data preprocessing and hardware compatibility of the model with OpenMV Cam H7. This proves that the resource-constrained device has the potential to generate comparable and even improved performance. Additionally, the results exhibit that smaller models, such as mAlexNet with a size of 19-39 KB showcase commensurate or even higher accuracy in contrast to bigger models, such as Mohan's model of size 137 KB.

Table 1. Accuracy of models.

Model	Raspberry Pi		OpenMV	
	Accuracy	Size	Accuracy	Size
mAlexnet224	94.78	39 KB	-	-
mAlexnet32	-	-	90.85	19KB
Mohan's model	87.04	137 KB	91.55	137 KB

Table 2. Accuracy of models under different conditions.

Model	*Day*	*Night*	*Snow*	*Rain*	*Obstacles*
mAlexnet224	98.26	87.85	97.91	26.56	32.50
mAlexnet32	98.26	82.29	92.01	54.69	47.50
Mohan's model	96.53	89.93	88.19	87.50	57.50

Table 3. Analysis of models for day-night-snow settings.

	Day		**Night**		**Snow**	
Model	FP	FN	FP	FN	FP	FN
mAlexnet224	3	2	29	5	0	6
mAlexnet32	4	1	49	5	19	4
Mohan's model	0	10	17	12	8	26

Table 2 encapsulates the key results obtained from our assessment, offering a comprehensive overview of the models' performance across 5 settings, such as Day, Night, Snow, Rain, and Obstacles. Each evaluation scenario, comprising Day-Night-Snow settings, encompasses results from two distinct angles and a total of 2592 parking slot instances in 162 parking lot configurations. Furthermore, the Rain and Obstacles settings specifically gauge the model's capability to discern objects beyond vehicles, treating them as negative instances.

It's worth noting that the accuracy across the Day-Night-Snow settings remains relatively consistent for all models. However, Mohan's model notably achieves significantly higher results in the Rain and Obstacles setting.

Given the binary classification nature of the problem, we can employ FP-FN analysis to assess the models' performance in Day-Night-Snow settings. This approach enables us to discern the inclinations

of each model towards a specific class, with FP representing false positives and FN representing false negatives.

Table 3 illustrates that decreased light intensity tends to bias the models towards a positive class, which indicates an occupied parking space. Notably, Mohan's model appears to be the least affected, while mAlexNet32 is the most influenced. Furthermore, it's apparent that mAlexNet32 consistently exhibits a tendency towards the positive class across all settings, albeit to varying degrees. Conversely, Mohan's model demonstrates a comparatively stronger inclination towards the negative class, which is an unoccupied space, particularly under normal lighting conditions.

7. Conclusion and possible future work

In conclusion, our senior project tackled the problem of inefficient urban parking by developing a parking occupancy detection system. Unlike other smart parking systems proposed in the literature that rely on the traditional approach of cloud computing—where video data is sent over a network to a distant server for processing—we focused on edge computing instead, shifting all processing to camera-equipped edge devices installed directly in the parking lot.

During the project, we adapted existing CNN models, specifically mAlexNet and Mohan's model, for efficient deployment on OpenMV and Raspberry Pi and conducted thorough evaluations of these models. Evaluations demonstrated promising results, proving that our system could potentially be used in real parking lots to inform drivers about parking availability. Not stopping at model development and deployment, a Telegram bot was also implemented to send parking information directly to end-users, ensuring that our solution covered all aspects of system implementation—from model training and deployment to user interaction.

For future improvements, exploring object detection algorithms like YOLO instead of image classification models could potentially yield even better results. Furthermore, to ensure the system works as well in real life as it did during tests, it should be validated in actual parking environments.

8. References

[1] F. M. Awan, Y. Saleem, R. Minerva, and N. Crespi, “A comparative analysis of machine/deep learning models for parking space availability prediction,” Sensors, vol. 20, no. 1, 2020.

[2] J. Ellis, A. Harris, N. Saquer, and R. Iqbal, “An analysis of lightweight convolutional neural networks for parking space occupancy detection,” pp. 261–268, 11 2021.

[3] D. Acharya, W. Yan, and K. Khoshelham, “Real-time image-based parking occupancy detection using deep learning,” Proceedings of the 5th Annual Conference of Research@Locate, Adelaide, Australia, vol. 2087, p. 33–40, 2018.

[4] N. Dan, “Parking management system and method,” Jan. 2002. U.S. Patent App. 10/066,215.

[5] Q. Wu, C. Huang, S.-y. Wang, W.-c. Chiu, and T. Chen, “Robust parking space detection considering inter-space correlation,” Multimedia and Expo, 2007 IEEE International Conference on, Jul 2007.

[6] C. G. Postigo, J. Torres, and J. M. Men´endez, “Vacant parking area estimation through background subtraction and transience map analysis,” IET Intelligent Transport Systems, vol. 9, p. 835–841, Nov 2015.

[7] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, “Car parking occupancy detection using smart camera networks and deep learning,” 2016 IEEE Symposium on Computers and Communication (ISCC), Jun 2016.

[8] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo, “Deep learning for decentralized parking lot occupancy detection,” Expert Systems with Applications, vol. 72, 10 2016.

[9] P. Mohan, A. Paul, and A. Chirania, A Tiny CNN Architecture for Medical Face Mask Detection for Resource Constrained Endpoints, pp. 657–670. 01 2021.

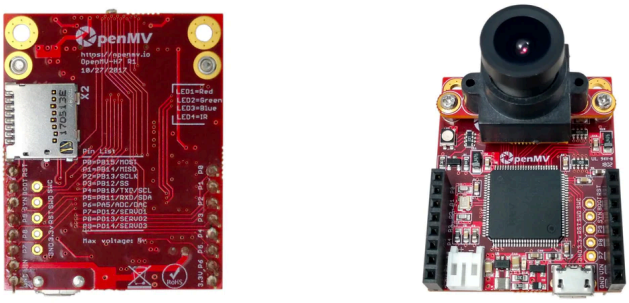
[10] “OpenMV Cam H7.” OpenMV.

[11] CNRPark, “Cnrpark: A smart parking system,” 2024. Accessed: 2024-03-22.

[12] C. Gronfier, K. P. Wright Jr, R. E. Kronauer, and C. A. Czeisler, “Entrainment of the human circadian pacemaker to longer-than-24-h days,” Proceedings of the National Academy of Sciences, vol. 104, no. 21, pp. 9081–9086, 2007.

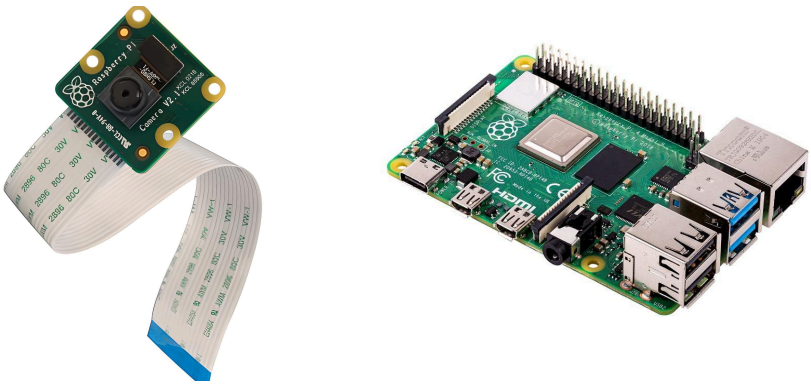
[13] J. Y. Suk and R. Walter, “Street lighting and public safety: New nighttime lighting documentation method,” URL: https://www. researchgate. net/publication/325273795 Street Lighting and Public Safety New Nighttime Lighting Documentation Method, 2018

8. Appendix



(a) Front Part (b) Back Part

Figure 1-2. OpenMV Cam H7.



(a) Raspberry Pi Model 4 B (b) Pi Camera

Figure 3-4. Raspberry Pi.

Figure 5. Overview of CNRPark+EXT divided into 2 classes: busy (1st row) and empty (2nd row). 3 weather scenarios are represented: sunny (1st column), rainy (2nd column), and overcast (3rd column).

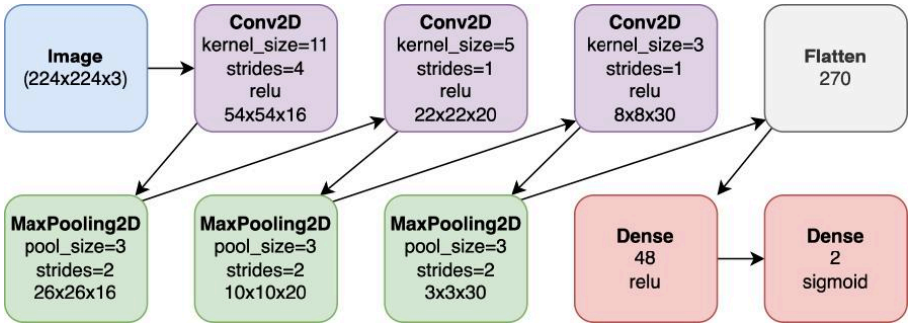


Figure 6. mAlexNet224 architecture.

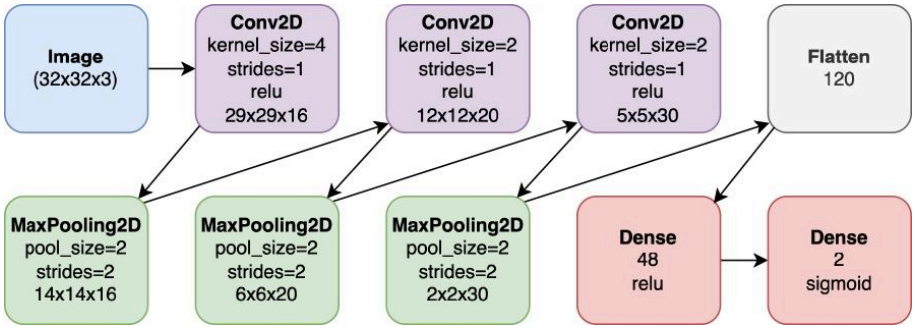


Figure 7. mAlexNet32 architecture.

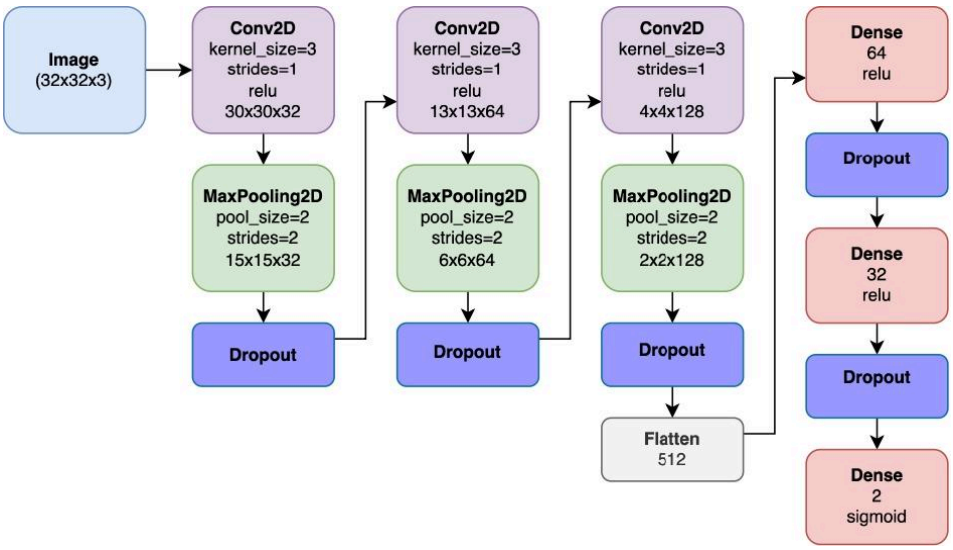


Figure 8. Mohan's architecture.

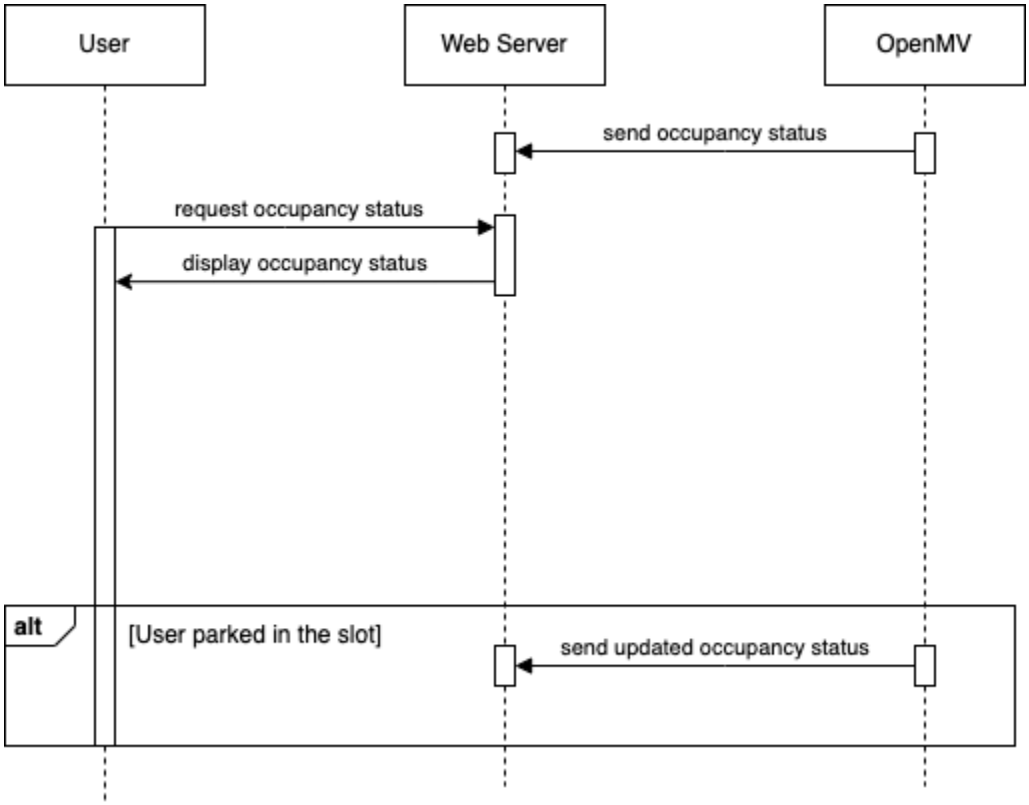


Figure 9. Sequence diagram of the system.

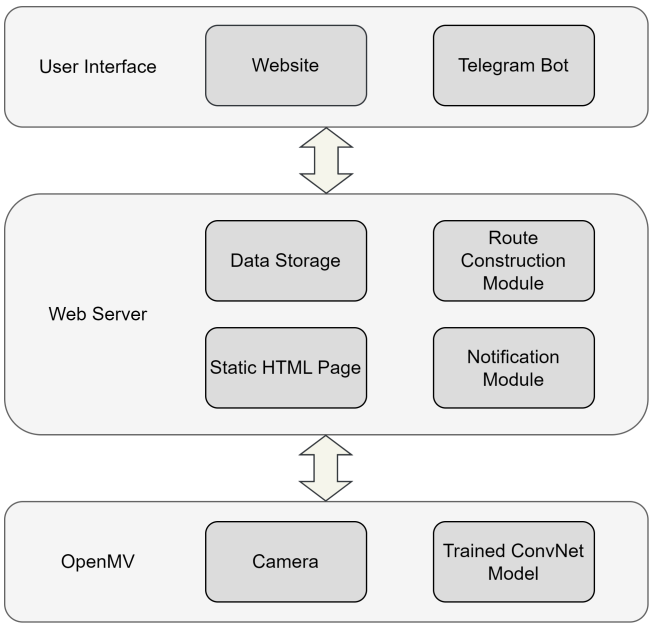


Figure 10. Layered view of the system.

Figure 11. Parking lot simulator with toy cars.

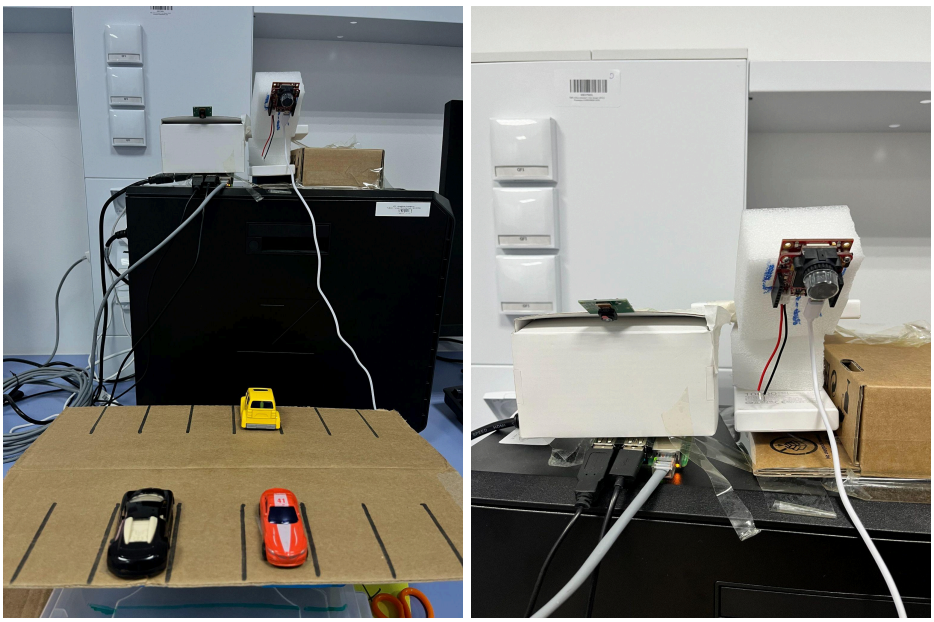


Figure 12. Full hardware setup with toy parking setting. Devices: Pi Camera the on left and OpenMV Cam H7 the on right.

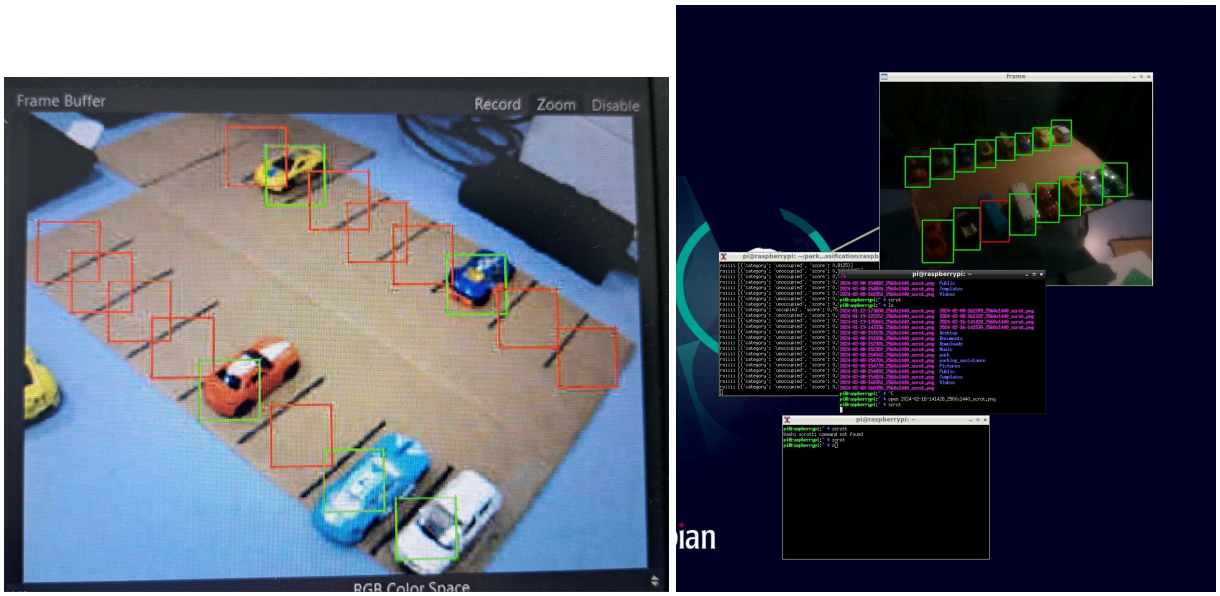


Figure 13. OpenMV IDE and Raspberry Pi desktop screenshots

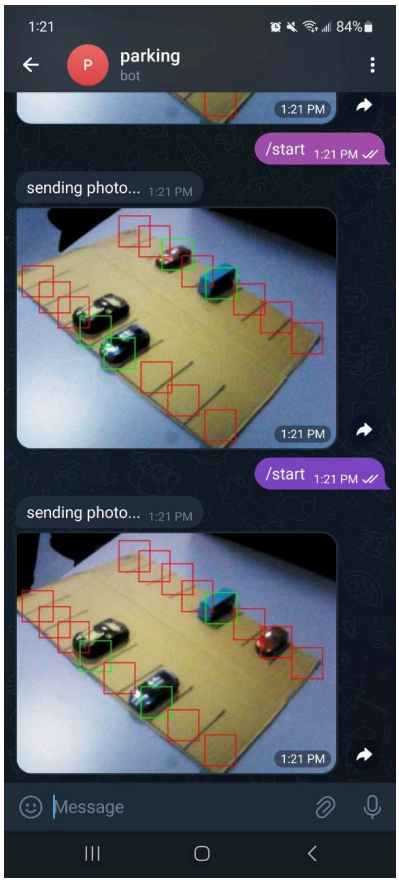


Figure 14. Telegram bot.

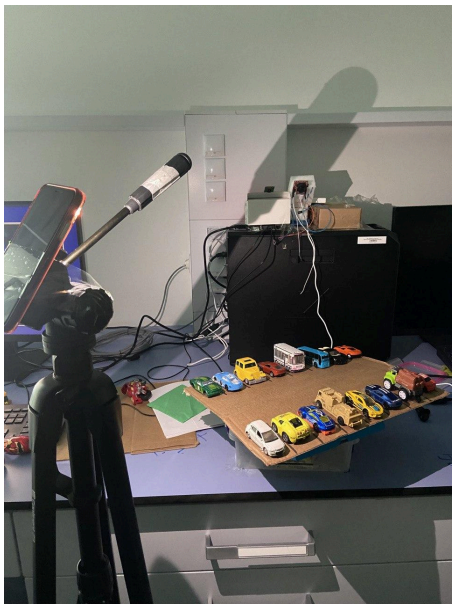


Figure 15. The simulation of varying light intensity on a parking lot.

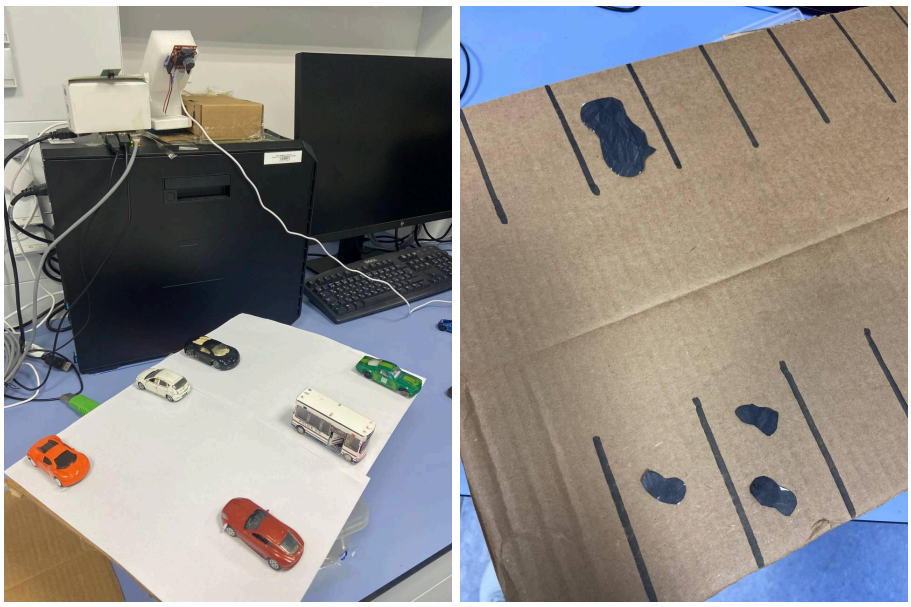


Figure 16. Parking lot under different weather conditions: snow cover and paddles.

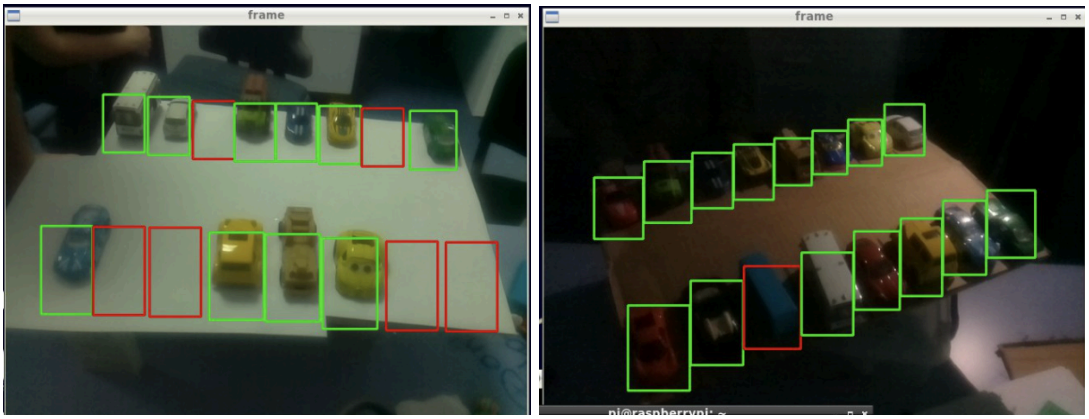


Figure 17. Parking lot from two different angles viewed by Raspberry Pi with Pi Camera.

Figure 18. Parking lot with obstacles: non-vehicle objects, such as humans and animals.