10 Years of Deep Learning for Vehicle Detection at a Smart Parking : What has Changed?

Gustavo P. C. P da Luz[1], **Gabriel Massuyoshi Sato**[1],
Tiago Godoi Bannwart[1], **Luis Fernando Gomez Gonzalez**[1],
Juliana Freitag Borin[1]

[1]Instituto de Computação (UNICAMP), Campinas – São Paulo – Brasil

{g271582, g172278, t215386}@dac.unicamp.br
{gonzalez, jufborin}@unicamp.br

Abstract. *Over the past decade, deep learning has transformed vehicle detection systems, particularly in smart parking applications. This work presents the evolution of a near real-time parking lot monitoring system at a university campus through multiple research and development iterations. The latest version incorporates the YOLOv11m model, converted to TensorFlow Lite, achieving a balanced accuracy of 98.65% on a dataset of 3,484 images, with an inference time of 8 seconds and updates sent to the parking totem every minute. The system integrates a Raspberry Pi 3 for edge inference, sending the number of available parking spaces to an InfluxDB database, while an ESP8266 receives this data and displays it on an LED panel at a totem. Earlier versions faced challenges such as hardware limitations, lack of documentation, and reliance on outdated deep learning models. The new version addresses these issues, improving detection accuracy, with enhanced fault tolerance, and a more robust totem structure. Additionally, a benchmark comparing 13 deep learning models for smart parking is introduced, along with software and hardware instructions for system replication.*

1. Introduction

The 2030 United Nations Agenda represents a global commitment by member countries, including Brazil, to promote economic, social, and environmental transformation towards sustainable development [Nations 2015]. Among the Sustainable Development Goals (SDGs) of this agenda, the 11th objective stands out, which includes creating inclusive, safe, resilient, and sustainable cities and communities.

Aligned with these goals, Unicamp has launched the Smart Campus initiative[1], aiming to transform the campus into a more intelligent, efficient, and sustainable environment by implementing Internet of Things (IoT) technologies. These technologies seek to optimize decision-making processes and improve both productivity and quality of life in the university environment. One of the proposals of Smart Campus is the implementation of smart parking systems. Given the high demand for parking spaces on the Unicamp campus, the objective of this system is to reduce the time spent searching for a space. It is estimated that up to 30% of urban traffic is related to vehicles searching for a parking space [Shoup 2021], especially in areas with high population density

[1]Smart Campus Unicamp

[Assemi et al. 2020, Rodríguez et al. 2024]. This assessment may also include contextual factors such as traffic patterns and occupancy rates [Dowling et al. 2017].

Effective parking management strategies not only reduce congestion and search times but also contribute to lowering greenhouse gas emissions and fuel consumption [Rodríguez et al. 2024]. Reducing parking search times is particularly impactful during peak hours, when the number of vehicles is highest, leading to significant reductions in carbon emissions compared to non-optimized environments [Paidi et al. 2022]. Furthermore, smart parking systems have the potential to improve the user experience and generate valuable data for urban planning and the development of more integrated and sustainable environments [de Moraes et al. 2024].

Traditional smart parking systems often rely on infrared, ultrasonic, or magnetic sensors, which are less suitable for outdoor scenarios and can be costly. An alternative approach is the use of vision-based solutions, which rely on cameras to capture images of parking lots and monitor availability. This approach is more cost-effective and easier to install and maintain, as a single camera can cover multiple parking lot spots. Despite of its advantages, deploying popular deep learning models on edge devices presents challenges, particularly in terms of computational efficiency. The field of edge intelligence explores the intersection between IoT and artificial intelligence. A common scenario is using pretrained models at large datasets and deploying them to devices such as a Raspberry Pi, keeping a local inference at the edge. A common research question when selecting a model to be deployed in a smart parking system is: What is the most suitable model that offers a good detection rate while fitting within the constraints of a limited edge device in terms of memory and inference time?

This work aims to compare the historical rounds of research and development of a near real-time monitoring system of the number of available parking spaces in the building of the Institute of Computing - 2 (IC-2) at Unicamp. The main contributions of this work are a benchmark consisting of 13 deep learning models for smart parking, including one achieving 98.65% balanced accuracy suitable for edge devices, and a comprehensive system integrating new and legacy codebases with hardware instructions for building a smart parking system[2].

The remainder of the paper is structured as follows: Section 2 reviews related work and the previous versions of the system, including architectures, functionalities, and limitations. Section 3 details the proposed system and its improvements. Section 4 presents the results obtained. Section 5 presents the lessons learned and difficulties faced during the development. Finally, Section 6 concludes the paper and suggests directions for future work.

2. Previous Systems and Related Work

As shown in Figure 1, since 2015, the project has undergone multiple rounds of research and development. We will first present the general concept of the system, followed by an explanation of each phase. The system utilizes a camera installed in the IC-2 parking lot at UNICAMP, which captures images and can cover the entire parking lot in a single frame, monitoring a total of 16 parking spaces. The captured data can be processed with

[2]https://github.com/discovery-unicamp/smartparking_unicamp/

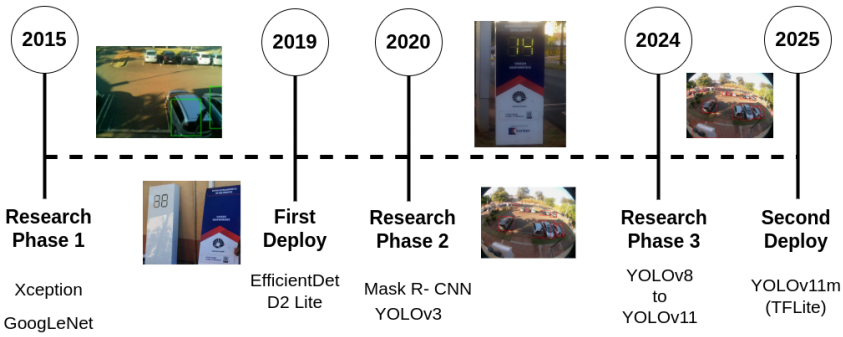

Figure 1. Research and Development Stages of the System.

inference performed either locally or remotely. Remote inference occurs when the image is transmitted over the network, and the inference is performed outside the device responsible for capturing the image. This approach raises concerns regarding both latency and privacy. To address these issues, we focus on performing neural network inference directly on the edge device, which consists of a Level 3 edge intelligence device, as defined by [Zhou et al. 2019]. Specifically, we use a model that has been trained in the cloud and then transferred to the device for inference. The inference determines the number of available parking spots, which is sent to an associated IoT platform for further analysis. Additionally, real-time parking availability is displayed on a totem located at the parking lot entrance, assisting drivers in finding open spaces more efficiently.

The edge device used since 2019 is a Raspberry Pi 3B+ with 1GB of memory and with increased SWAP memory configuration, designed for low-cost applications that require processing power and connectivity. It features a 32-bit ARM architecture and is well-suited for embedded projects, such as real-time monitoring and control systems. The model used for inference, along with other software components, has evolved throughout the project. In the following sections, we will discuss the key changes made in each phase , as well as highlight other works that followed these changes.

2.1. Research Phase 1 (2015 - 2019)

The first research phase began around 2015 with the challenge of implementing a smart parking system based on cameras to help the university community identify available parking spots. At that time, deep learning was gaining significant attention worldwide, and object detection was a rapidly evolving field, driven by competitions like ImageNet, which encouraged research teams to develop more advanced detection models. Convolutional Neural Networks (CNNs) significantly improved detection performance by enabling automatic feature extraction through deep learning. Before CNNs, object detection relied on classical methods that depended on handcrafted features and algorithms, which struggled with variations in lighting, occlusion, and other challenges. The adoption of deep neural network models helped address these issues, leading to more robust and accurate detection systems.

Inspired by this, the research team began experimenting with the GoogleLeNet model [Szegedy et al. 2015], which won the 2014 ImageNet challenge. This model extends one of the foundational CNN architectures, enabling efficient training of deeper networks and incorporating multiple filter sizes to capture different feature scales. How-

ever, using a pre-trained GoogleLeNet model did not yield satisfactory results, leading the team to fine-tune it with images of the parking lot using the now-deprecated tool TensorBox[3]. To annotate the dataset, a teacher-student distillation approach was used. A pre-trained Xception instance segmentation network [Chollet 2017], which was too large for deployment on an edge device, served as a teacher, generating bounding boxes around vehicles. These were then used to train the student model (GoogleLeNet). This first research phase laid the foundation for subsequent developments, and more details can be found in a presentation[4] delivered at an international conference focused on real-world machine learning applications. Other smart parking solutions also trained LeNet to detect vehicles such as [Amato et al. 2017, Nyambal and Klein 2017].

2.2. First Deploy (2019)

After the first research phase, the team faced the challenge of deploying deep learning models on edge devices, as most state-of-the-art architectures were impractical for embedded IoT systems. A key turning point was the adoption of the Single Shot Detector (SSD) EfficientDet [Tan et al. 2020], a model developed by Google that builds upon traditional networks and used for vehicle detection in works such as [Greco et al. 2021].

The use of SSD marked a significant milestone in object detection for edge devices. Unlike two-stage detectors, which generate region proposals and then classify and refine them in a second stage, SSD models perform a single forward pass of the neural network to predict bounding boxes and class labels simultaneously. This approach significantly improves efficiency, making it more suitable for real-time applications on resource-constrained devices.

Another key factor in deployment is the option to convert the models for a lighter version using TensorFlow Lite (TFLite). Introduced in 2017, TFLite optimizes the model for low-power devices by applying quantization and reducing memory footprint. These optimizations led to the selection of the EfficientDet-D2 Lite model, which became the first model successfully deployed in the parking lot.

2.3. Research Phase 2 (2020 - 2024)

The second stage of the project's research included evaluating more advanced models that had not been tested before, such as versions of the YOLO network and Mask R-CNN. The YOLO (You Only Look Once) models stood out due to their superior inference speed compared to other CNN's since YOLO is also a single-pass detector. The first version YOLOv1 was inspired by GoogleLeNet and introduced by Redmon in 2016 [Redmon et al. 2016]. The same creator evolved the network until YOLOv3 [Redmon and Farhadi 2018], introducing improvements in multi-scale detection and feature fusion. The results of using a pre-trained YOLOv3 model were reported at [Baggio et al. 2020] and despite achieving good results for that time, this phase of research did not result in a system deployment. Other smart parkings based on YOLOv3 are presented by [Abbas et al. 2023, Satyanath et al. 2023] , where YOLO is shown to be a more suitable alternative than Mask R-CNN for edge-based deployments.

[3]TensorBox

[4]PAPIs.io LATAM 2018 - Full Conference Day 1

Even though SSD-based models were the most suitable for the context of the project, experiments were conducted with the two-stage detector Mask R-CNN [He et al. 2017], which was the state-of-the-art at the time. Mask R-CNN is built with a ResNet backbone and extends Region-based CNNs by adding a mask head to predict pixel-level segmentation masks for each detected object. The results were promising, but the model proved to be unsuitable for edge devices. Therefore, the experiments were conducted to compare the performance of the two different families of detectors.

2.4. Research Phase 3 (2024)

In 2024, several new versions of YOLO models were released, each introducing optimizations in feature extraction, more efficient backbones, and attention mechanisms to enhance accuracy and improve detection speed. Specifically, the improvements included anchor-free detection (YOLOv8), programmable gradient information (YOLOv9), position-sensitive attention (YOLOv10), and enhanced feature extraction (YOLOv11). A comparison was conducted between several versions of pre-trained YOLO models, from YOLOv8 to YOLOv11, across six different devices, along with two image processing methods. The results were reported in [da Luz et al. 2024a], and these findings helped guide the selection of the current model architecture. As these models were released recently, only a few other works such as [Doshi et al. 2024, An et al. 2024] evaluated them for vehicle detection at smart city related systems.

2.5. Second Deploy (2025)

Based on tests performed in [da Luz et al. 2024a], the YOLOv11 and YOLOv9 models achieved the best balanced accuracies, with YOLOv11 proving to be faster, which led to its selection for deployment. After selecting the model version, the next step was to determine the appropriate model size, as YOLO models offer multiple variants with different numbers of parameters. Each variant is designed to suit different processing capabilities, balancing accuracy and computational efficiency. Through device overhead testing, which measured CPU usage, memory consumption, and prediction performance, the YOLOv11m variant converted to TFLite was identified as the optimal choice. To convert to TFLite, we followed the step-by-step guide provided by Ultralytics[5]. As previously observed with EfficientDet models, this conversion is beneficial for resource-constrained edge devices and significantly reduces application latency [Verma et al. 2021].

3. Proposed System

Some limitations were identified in the previously deployed system. The primary issue was its discontinuation due to the termination of a partnership with a company providing data support. Additionally, the LED panel wires were found to be oxidized, and there was no documentation of the solution. The Region Of Interest (ROI) selection method, which relied on image cropping, was functional but lacked precision. Furthermore, the system used the EfficientDet-D2 Lite model, which could benefit from an updated model with higher accuracy. Based on these limitations, several improvements were implemented. This section compares the solutions using the typical layers of IoT projects.

Figure 2 shows that the layers were generally maintained, with some modifications. The first change was in the data support layer, which was previously an IoT platform

[5]A Guide on YOLO Model Export to TFLite for Deployment

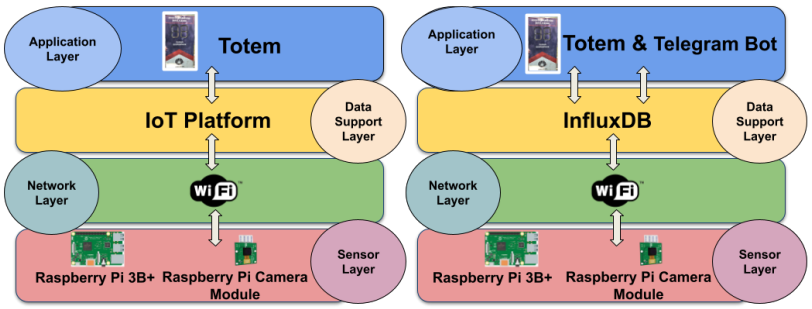

Figure 2. IoT layers of previous systems vs proposed system.

from a partner company but was later replaced with the time series database InfluxDB. Additionally, a Telegram bot functionality was introduced as part of the application layer. In the sensor layer, a Raspberry Pi is connected to its standard camera module, which captures photos, processes them, and transmits the number of available parking spots to InfluxDB over Wi-Fi. Finally, applications such as the totem and the Telegram bot display this information to end users and stakeholders.

Figure 3 illustrates the system model, depicting the complete data flow from the moment a new image is captured to when the information is displayed on the LED panel. The process begins on the Raspberry Pi, where a Python script establishes connections with InfluxDB and Redis, a local database that temporarily stores data in a queue before transmitting it via Wi-Fi. A custom mask is applied, enabling pixel-wise selection to count vehicles parked within the designated region during post-processing.

After reading the mask, a loop is initiated, in which the photo of the parking lot is captured and the image is preprocessed for the machine learning model, with normalization and resizing. Resizing is done from 764x1024 pixels to match the input shape of pre-training. Specifically, YOLOv3 uses an input shape of 416x416, EfficientDet-D2 Lite uses 448x448, Mask R-CNN uses 1024x1024, and YOLOv8 to v11 use 640x640. The input shape significantly impacts both model size and performance. After the inference is performed, Non-Maximum Suppression (NMS) is applied as post-processing, ensuring that only the most confident detections are retained, eliminating duplicate predictions for the same object. After NMS, using a method detailed in [da Luz et al. 2024a], the system counts the vehicles that were detected within the ROI and this number is sent to Redis, where it is stored temporarily before being forwarded to InfluxDB. The number of available parking spaces is calculated by subtracting the detected vehicle count from the maximum capacity of 16 spaces.

With the number of available spaces in the database, various applications can consume this data. On the totem's side, an ESP8266 periodically requests the latest data from the database and sends it to the LED panel for display. Similarly, a Telegram bot queries the database and notifies users of any delays in data transmission by comparing the current time with the timestamp of the last update.

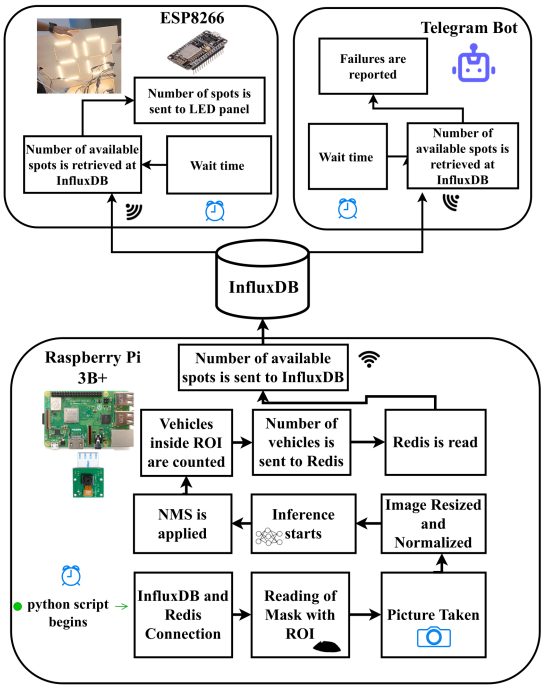

Figure 3. Data flow of the number of parking spots from the sensor layer to the application layer. Component image sources: ESP8266, Raspberry Pi 3B+, and Raspberry Pi Camera Module.

The system has been enhanced with several improvements. The EfficientDet-D2 Lite model has been replaced with YOLOv11m, converted from PyTorch to TensorFlow Lite (TFLite) in its float16 version, improving detection accuracy and computational efficiency. To ensure long-term reliability, a scheduled reboot mechanism has been implemented, and remote access has been enabled via SSH. Existing features were maintained, including the use of a watchdog timer on the ESP8266, the Supervisor process control system on the Raspberry Pi, Redis for data queuing, and the Unicamp network specific designed for IoT devices. Additionally, a Telegram bot has been added to provide real-time notifications and system status updates, enhancing user interaction and monitoring capabilities.

Regarding the operation of the totem, the previous schematic was maintained but the LED panel has been upgraded with higher-brightness LEDs for improved readability and a more robust, weather-resistant base.

Figure 4 shows an overview of how the circuit in the totem works. It was built with two panels, one on each side so that the number of available spaces is visible regardless of which side it was viewed from. The microcontroller periodically retrieves the last data sent to InfluxDB and displays it on the panel. The entire circuit is powered by a 12V power supply. However, since the development board and the multiplexers [6] operate at 5V, an LM2596 voltage regulator is used to step down the voltage. The power lines are indicated by the red and black tracks as shown in Figure 4.

[6]Multiplexer datasheet

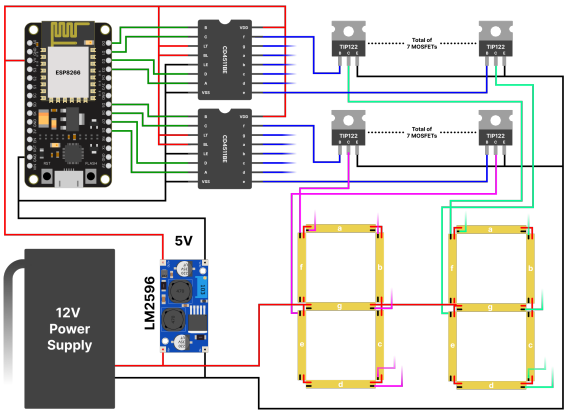

Figure 4. Schematic of how the totem works.

To control each of the 14 segments of the panel individually, the specialized multiplexer CD4511BE was used. This component decodes its input into signals for a 7-segment display. Thus, to display the correct digits on the panel, the development board adjusts its I/O pins to set the corresponding binary values required by the multiplexer, as illustrated by the dark green traces in Figure 4. Since the multiplexer operates at a different voltage than the LEDs, TIP122 MOSFETs were used to switch the segments on and off. Additionally, to ensure visibility from both the front and back of the totem, the circuit was duplicated for the second panel. This duplication includes the MOSFETs and the two digits shown in the figure, effectively mirroring the entire section of the circuit.

4. Results and Discussion

4.1. Model performance

We developed a benchmark with 13 different models that have been considered in the past years by the research team members and provided the balanced accuracy, inference time and model size to discuss how the project evolved. The dataset used was previously collected by the project members and consists of 4477 images containing various scenarios including day, night, rain, sun, and other variations. The dataset is imbalanced, with fewer images containing vehicles than empty spaces. To address this, balanced accuracy was used as the evaluation metric, as it is better suited for imbalanced datasets by considering performance across both classes. The test set was filtered to include only images with at least one vehicle to ensure a realistic evaluation, removing 22% of the dataset, resulting in a final test set of 3484 images.

To maintain a consistent methodology at the evaluation process, we followed the methodological steps depicted in [da Luz et al. 2024a]. Each image was resized to the input dimensions required by the respective model according to the pre-training phase shape. We employed the same post-processing method to select the ROI. Prior to that, different pre-processing methods were used, such as cropping the regions and using a mask to change the pixel values at the input image.

Figure 5 shows how each model performs and provides a comparison of the trade-offs between balanced accuracy and the size of the model. Table 1 is complementary reporting the exact values of balanced accuracy and model size, and also the inference

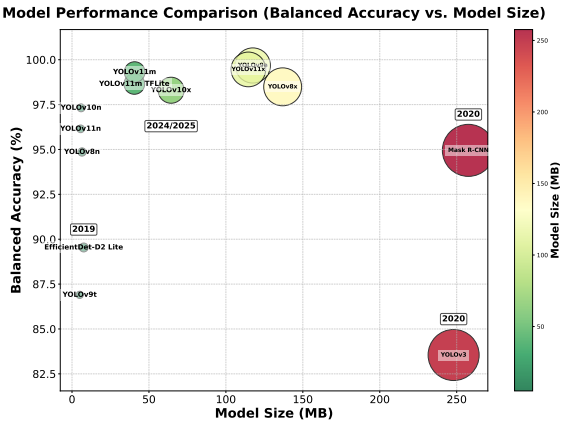

Figure 5. Evaluation of the models considered for a smart parking project in terms of model size and balanced accuracy.

Model	Bal. Acc. (%)	Inf. Time (s)[1]	Size (MB)
YOLOv9e[2]	99.68	92 ± 9.3	117.5
YOLOv11x[2]	99.46	46 ± 7.4	114.6
YOLOv11m	99.31	11 ± 2.3	40.7
YOLOv11m (TFLite)	98.65	8 ± 0.62	40.5
YOLOv8x[2]	98.49	28 ± 0.7	136.9
YOLOv10x[2]	98.30	24 ± 0.2	64.4
YOLOv10n[2]	97.32	2 ± 0.03	5.9
YOLOv11n[2]	96.15	2 ± 0.02	5.6
Mask R-CNN	94.95	–	257.6
YOLOv8n[2]	94.86	2 ± 0.06	6.5
EfficientDet-D2 Lite	89.54	1 ± 0.11	7.6
YOLOv9t[2]	86.90	2 ± 0.02	5.0
YOLOv3	83.55	9 ± 0.25	248.0

Table 1. Model performance ranked by balanced accuracy.

[1] Inference time for Mask R-CNN could not be measured due to the memory constraint imposed by the 1 GB RAM capacity of the Raspberry Pi 3B+.
[2] Values measured at the same hardware at a previous work [da Luz et al. 2024a].

time of each model measured in the edge device used in the parking lot. The EfficientDet D2 Lite model deployed in 2019 showed to be a great choice since it reaches 89.54% balanced accuracy, with the lowest inference time of one second and a small model size. Contrasting with that, it can be seen that other versions considered in the second research phase until 2024 have higher model sizes. Also, in the case of YOLOv3, a lower balanced accuracy was observed. These factors partly justifies the lack of most recent deployments of the system.

Moving to the last versions of YOLO models released (8 to 11) tested in 2024, it is possible to note that the smaller version of the models appear consistently at the left side of Figure 5, with YOLOv10n and YOLOv11n being the ones that reach the highest balanced accuracy with smallest model size. The largest variants of the YOLO models appear in the top middle of the figure, reaching the highest possible balanced accuracies (up to 99.68%), however with slow inference times and large model sizes. That turns the use of such models possible but not optimal for edge devices, since they drain more resources and may influence in the health of the hardware used.

To address these conditions, YOLOv11m (TFLite) achieves 98.65% balanced accuracy with a medium-sized model of 40.5 MB. Compared to the previously deployed EfficientDet D2 Lite model, YOLOv11m (TFLite) shows an improvement of over nine percentage points in balanced accuracy, with an inference time of 8 seconds. Compared to the original YOLOv11m model, the TFLite version has an balanced accuracy only 0.66% lower, but it is more suitable for edge devices and can be executed on 32-bit OS.

The increase in vehicle detection accuracy was expected to come at the cost of slower inference and a larger model size. However, this increase does not overload the Raspberry Pi 3, and the current inference time remains suitable for the parking lot's low turnover rate. Given that the totem display updates every minute, the system's performance is well within acceptable limits.

By performing a qualitative assessment, we can see the comparison between the

models from a visual perspective.

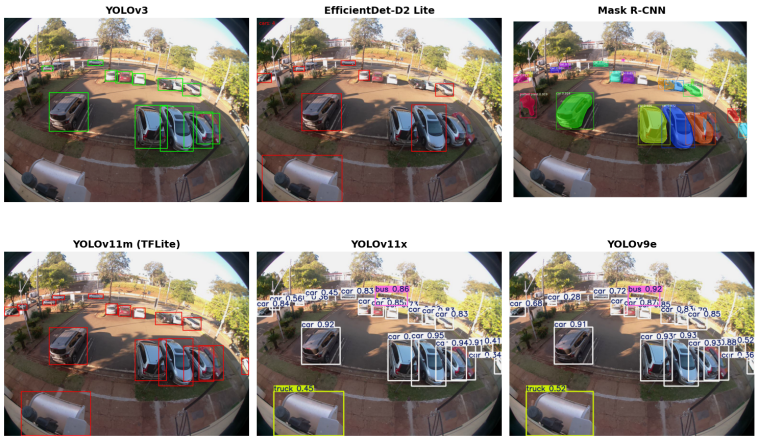

Figure 6. Qualitative analysis of the evolution of the models throughout the years.

Figure 6 illustrates a challenging scenario in the parking lot, where cars are exposed to direct sunlight. As shown, some models struggle with detection under these conditions. The top row represents models tested up until 2024, with Mask R-CNN and YOLOv3 demonstrating better detection than EfficientDet D2 Lite in this specific case, though performance may vary depending on the challenge. The bottom row highlights the evolution of YOLO models in terms of generalization capabilities, with YOLOv9e and YOLOv11x emerging as the best-performing models tested to date. Despite the challenging lighting conditions, the YOLOv11m (TFLite) model selected for deployment achieves a good detection rate.

4.2. System performance

To validate the system's functionality, a bench test was conducted in a controlled laboratory environment. A video demonstrating the real-time operation of the system is available in the project's GitHub repository available in the Section Reproducibility Statement. Furthermore, instructions for implementing the Telegram bot, along with a demonstration, are also provided. After the initial rounds of lab testing, the Raspberry Pi was reinstalled in the parking lot in December 2024.

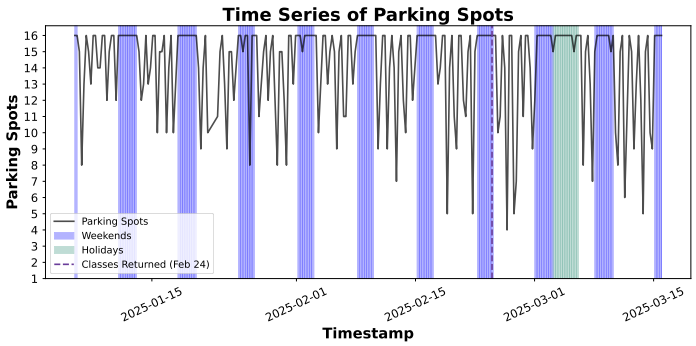

Figure 7. Data sent and received by InfluxDB in the first three months of 2025.

Figure 7 illustrates real-world collected data of parking occupancy for the first three months of 2025. This data was recorded by the Raspberry Pi and sent at a one-

minute interval to InfluxDB. From the moment the picture is collected to the display in the totem, total latency is approximately one minute. This interval was chosen based on the parking lot turnover but could be lower, as the inference process takes only 8 seconds. Weekends and holidays show deviations to regular weekdays with more parking spots. The dashed purple line marks the return of academic term, which coincides with a potential shift in parking behavior. This suggests that the parking demand is influenced by the current schedule of the university, and can bring insights for traffic flow optimization inside the campus, specially as more parking lots benefits with similar systems.

5. Lessons Learned and Challenges

During the development of this project, some restrictions and difficulties were encountered. The decision to maintain the existing Raspberry Pi 3 device imposed constraints on the software design, particularly due to the 32-bit Raspberry Pi OS operating system. This restricted the availability of modern libraries, implying on using the TensorFlow Lite Runtime interpreter. Initial attempts to use a Raspberry Pi 0 proved to be possible but impractical. The Raspberry Pi 3B+, with 1GB of memory and with increased SWAP memory configuration, represents our minimum hardware tested for the application. This finding highlights the need for careful hardware selection in edge computing applications, where computational resources are often limited. This experience reveals the critical need for documenting software versions, dependencies, and configurations to ensure reproducibility, especially when working with legacy systems.

Regarding the totem circuit side, ensuring reliable electrical connections was a challenge. Poor contact wires between the wires and MOSFETs led to failures, which resulted in new rounds of soldering and ultimately in manufacturing a Printed Circuit Board (PCB). Additionally, the camera's placement in a high, inaccessible location introduced logistical and bureaucratic challenges, as access required coordination with the maintenance team. This difficulty emphasized the importance of designing systems with fault-tolerant mechanisms to minimize the need for frequent maintenance. Another difficulty is repositioning the device after removal. Since a handcrafted mask is used to select the ROI for vehicle counting, it is unlikely that the camera will be positioned in the exact previous view, requiring the creation of a new mask in an image manipulation program.

6. Conclusions and Future Work

In this work, we provided a benchmark to evaluate the performance of 13 established deep learning models within the smart parking system at the Unicamp Institute of Computing. Additionally, the current system model was assessed, achieving a balanced accuracy of 98.65%, with an inference time of 8 seconds and a model size of 40.5 MB. This represents an improvement of over nine percentage points in balanced accuracy compared to the previously deployed model, while maintaining a suitable inference time. Furthermore, the software and hardware of the solution were updated, facilitating the development of other smart parking or object detection-related systems. The challenges encountered over a decade of research and development were also reported and documented, contributing to future advancements in the field.

As future work, it is possible to add new options of deep learning models and edge devices to the benchmark as the state of the art evolves, such as

YOLOv12 [Tian et al. 2025], Transformers based real time models [Huang et al. 2024, Peng et al. 2024], and devices such as Raspberry Pi 5 and NVIDIA Jetson. In addition to the mentioned devices, it is possible that the system could be used together with the Unicamp project to reuse TV Boxes seized by the Federal Revenue Service as edge devices, instead of using a Raspberry Pi, which would further contribute to the SDGs, in line with recent work that studied the feasibility of this use in a similar application [da Luz et al. 2024b, Sato et al. 2024]. Such a change would imply modifying the box that encloses the device and replacing the current camera with a USB camera module.

In the data support layer, InfluxDB could be replaced by an IoT platform that offers broader functionalities. This platform could be integrated using standardized data exchange protocols such as NGSI-LD, along with software frameworks built upon them like FIWARE[7] and Smart Data Models[8] for the Smart City domain, enhancing interoperability across various smart city and campus solutions.

At the application layer, the bot can be enhanced with daily statistics on the number of inferences made and vehicles detected. Additionally, it could support user requests for real-time parking availability, helping users plan their departure from home more efficiently. This real-time availability data could also be integrated into the Unicamp Services App, which already offers various features to enhance campus life. While the smart parking system at the Institute of Computing currently manages only 16 spaces, the approach can be extended to more complex scenarios, such as the Unicamp rectory parking lot, which operates with a four-camera system and manages over 100 parking spots.

Acknowledgements

The authors would like to thank everyone who historically contributed to this project. This project was supported by CAPES (process 88887.999360/2024-00), CNPq (process 308840/2020-8 and 131653/2023-7), by the Brazilian Ministry of Science, Technology and Innovations, with resources from Law nº 8,248, of October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex and published Arquitetura Cognitiva (Phase 3), DOU 01245.003479/2024 -10, and by FAPESP (process 2023/00811-0).

Reproducibility Statement

The code used in the experiments, along with instructions to build the hardware, is publicly available on GitHub[9]. A small subset of images from the CNRPark-EXT dataset [Amato et al. 2017] is used for demonstration purposes only. Downsampling was performed during data collection to minimize identifiable content. Nonetheless, the full dataset is stored with restricted access at the Institute of Computing, Unicamp, in line with a conservative approach to potential privacy issues. While license plates are not visible, any that do appear in this work are blurred as a precautionary privacy measure.

References

[Abbas et al. 2023] Abbas, Q., Ahmad, G., Alyas, T., Alghamdi, T., Alsaawy, Y., and Alzahrani, A. (2023). Revolutionizing urban mobility: Iot-enhanced autonomous parking solutions with transfer learning for smart cities. *Sensors*, 23(21):8753.

[7] FIWARE

[8] Smart Data Models

[9] https://github.com/discovery-unicamp/smartparking_unicamp/

[Amato et al. 2017] Amato, G., Carrara, F., Falchi, F., Gennaro, C., Meghini, C., and Vairo, C. (2017). Deep learning for decentralized parking lot occupancy detection. *Expert Systems with Applications*, 72:327–334.

[An et al. 2024] An, R., Zhang, X., Sun, M., and Wang, G. (2024). Gc-yolov9: Innovative smart city traffic monitoring solution. *Alexandria Engineering Journal*, 106:277–287.

[Assemi et al. 2020] Assemi, B., Baker, D., and Paz, A. (2020). Searching for on-street parking: An empirical investigation of the factors influencing cruise time. *Transport Policy*, 97:186–196.

[Baggio et al. 2020] Baggio, J. V., Gonzalez, L. F., and Borin, J. F. (2020). Smartparking: A smart solution using deep learning. Technical report, Instituto de Computação - UNICAMP.

[Chollet 2017] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258.

[da Luz et al. 2024a] da Luz, G. P., Sato, G. M., Gonzalez, L. F. G., and Borin, J. F. (2024a). Smart parking with pixel-wise roi selection for vehicle detection using yolov8, yolov9, yolov10, and yolov11. *arXiv preprint arXiv:2412.01983*.

[da Luz et al. 2024b] da Luz, G. P. P., Sato, G. M., Gonzalez, L. F. G., and Borin, J. F. (2024b). Repurposing of tv boxes for a circular economy in smart cities applications.

[de Moraes et al. 2024] de Moraes, P. A., Pisani, F., and Borin, J. F. (2024). Smart university: A pathway for advancing sustainable development goals. *Internet of Things*, page 101246.

[Doshi et al. 2024] Doshi, Y., Shah, K., Katre, N., Sawant, V., and Correia, S. (2024). Comparision of yolo models for object detection from parking spot images. *Educational Administration: Theory and Practice, 30 (4)*, pages 10401–10411.

[Dowling et al. 2017] Dowling, C., Fiez, T., Ratliff, L., and Zhang, B. (2017). How much urban traffic is searching for parking? *arXiv preprint*.

[Greco et al. 2021] Greco, A., Saggese, A., Vento, M., and Vigilante, V. (2021). Vehicles detection for smart roads applications on board of smart cameras: a comparative analysis. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):8077–8089.

[He et al. 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

[Huang et al. 2024] Huang, S., Lu, Z., Cun, X., Yu, Y., Zhou, X., and Shen, X. (2024). Deim: Detr with improved matching for fast convergence. *arXiv preprint arXiv:2412.04234*.

[Nations 2015] Nations, U. (2015). Transforming our world: The 2030 agenda for sustainable development. *New York: United Nations, Department of Economic and Social Affairs*, 1:41.

[Nyambal and Klein 2017] Nyambal, J. and Klein, R. (2017). Automated parking space detection using convolutional neural networks. In *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech)*, pages 1–6. IEEE.

[Paidi et al. 2022] Paidi, V., Håkansson, J., Fleyeh, H., and Nyberg, R. G. (2022). Co2 emissions induced by vehicles cruising for empty parking spaces in an open parking lot. *Sustainability*, 14(7):3742.

[Peng et al. 2024] Peng, Y., Li, H., Wu, P., Zhang, Y., Sun, X., and Wu, F. (2024). D-fine: redefine regression task in detrs as fine-grained distribution refinement. *arXiv preprint arXiv:2410.13842*.

[Redmon et al. 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

[Redmon and Farhadi 2018] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[Rodríguez et al. 2024] Rodríguez, A., Alonso, B., Moura, J. L., and dell'Olio, L. (2024). Analysis of user behavior in urban parking under different level of information scenarios provided by smart devices or connected cars. *Travel Behaviour and Society*, 37:100847.

[Sato et al. 2024] Sato, G., Luz, G., Gonzalez, L., and Borin, J. (2024). Reaproveitamento de tv boxes para aplicação de contagem de pessoas na borda em cidades inteligentes. In *Anais do VIII Workshop de Computação Urbana*, pages 197–209, Porto Alegre, RS, Brasil. SBC.

[Satyanath et al. 2023] Satyanath, G., Sahoo, J. K., and Roul, R. K. (2023). Smart parking space detection under hazy conditions using convolutional neural networks: a novel approach. *Multimedia Tools and Applications*, 82(10):15415–15438.

[Shoup 2021] Shoup, D. (2021). Pricing curb parking. *Transportation Research Part A: Policy and Practice*, 154:399–412.

[Szegedy et al. 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[Tan et al. 2020] Tan, M., Pang, R., and Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10781–10790.

[Tian et al. 2025] Tian, Y., Ye, Q., and Doermann, D. (2025). Yolov12: Attention-centric real-time object detectors. *arXiv preprint arXiv:2502.12524*.

[Verma et al. 2021] Verma, G., Gupta, Y., Malik, A. M., and Chapman, B. (2021). Performance evaluation of deep learning compilers for edge inference. In *2021 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*, pages 858–865. IEEE.

[Zhou et al. 2019] Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., and Zhang, J. (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762.