Development of a system to analyze the utilization of a bicycle parking space

Part 1

T3100 / 5. Semester

of the Electrical Engineering program at the Dualen Hochschule Baden-Württemberg Stuttgart

Juan Sebastian Mejia Arenas

Delivery date: 21.01.24

Processing period:	6 Weeks
Matriculation number/ course:	9654407/ TEL21GR3
Training company:	Coperion GmbH, Stuttgart
Supervisor of the project:	M. Sc. Jerg Pfeil

Declaration

in accordance with§ 5 (3) f the "Study and Examination Regulations DHBW

Technik" dated 22.

September 2011.

I have written this thesis independently and have not used any sources or aids other than those sources and aids other than those specified.

Stuttgart, 15.01.2024

Place, date

Sebastian Mejía

Signature

Abstract

This project revolves around the development of an object detection system for a bicycle parking lot situated in front of the Dual University Baden-Württemberg "DHBW" in Stuttgart. With 288 outdoor parking spaces, the lot is monitored by a camera connected to a Raspberry Pi, capturing video sequences. The primary objective is to create a robust object detection system capable of identifying bicycles and motorcycles using the parking lot. This system aims to facilitate statistical analyses of parking lot usage under varying weather conditions and throughout the year.

The project delves into the evaluation of diverse deep convolutional network models designed for object recognition and localization in images or videos. Prioritizing contemporary models, the study leverages the latest advancements in machine learning and computer vision to assess their performance, particularly in resource-constrained environments like the Raspberry Pi.

Customization of a model through training with a specific image set is a critical aspect. The project selects a suitable training set and optimizes training parameters to ensure reliable detection. Post-training evaluation compares results to identify an effective model aligned with the project's objectives.

In the initial project phase, a training dataset was selected, and convolutional network models were systematically evaluated for object detection in our bicycle parking lot scenario. The outcome led to the selection of the YOLOv5n6 model. In the upcoming phase, there are plans to further develop and optimize the chosen model, including the conversion to TensorFlow Lite for Raspberry Pi deployment and exploration of post-detection techniques to enhance reliability. Threshold adjustments will fine-tune detection results, ensuring a robust object detection system for subsequent statistical analysis and graphical representation of parking lot usage data.

Key words: Object Detection, Convolutional Neural Networks, YOLO, Raspberry Pi, Parking Lot Analysis, Computer Vision.

Kurzfassung

Dieses Projekt konzentriert sich auf die Entwicklung eines Objekterkennungssystems für einen Fahrradparkplatz vor der Dualen Hochschule Baden-Württemberg "DHBW" in Stuttgart. Mit 288 Außenparkplätzen wird der Parkplatz von einer Kamera überwacht, die mit einem Raspberry Pi verbunden ist und Videosequenzen aufzeichnet. Das Hauptziel besteht darin, ein robustes Objekterkennungssystem zu schaffen, das in der Lage ist, Fahrräder und Motorräder auf dem Parkplatz zu identifizieren. Dieses System soll statistische Analysen der Parkplatznutzung unter verschiedenen Wetterbedingungen und im Laufe des Jahres erleichtern.

Das Projekt untersucht verschiedene Modelle von tiefen Faltungsnetzen, die für die Objekterkennung und -lokalisierung in Bildern oder Videos entwickelt wurden. Mit Priorität für zeitgenössische Modelle nutzt die Studie die neuesten Fortschritte im maschinellen Lernen und in der Computer Vision, um deren Leistung zu bewerten, insbesondere in ressourcenbeschränkten Umgebungen wie dem Raspberry Pi.

Die Anpassung eines Modells durch Training mit einem spezifischen Datensatz ist ein entscheidender Aspekt. Ein geeigneter Trainingsdatensatz wird ausgewählt, und Trainingsparameter werden optimiert, um eine zuverlässige Erkennung zu gewährleisten. Die Post-Training-Bewertung vergleicht die Ergebnisse, um ein effektives Modell zu identifizieren, das mit den Zielen des Projekts übereinstimmt.

In der ersten Projektphase wurden verschiedene Modelle für die Objekterkennung auf dem Fahrradparkplatzsystem bewertet, wobei das YOLOv5n6-Modell ausgewählt wurde. Die nächste Phase plant die Optimierung und Umstellung auf TensorFlow Lite für den Raspberry Pi. Durch Anpassungen der Schwellenwerte wird die Erkennung weiter verbessert, um ein robustes System für die statistische Analyse und grafische Darstellung der Parkplatznutzung zu schaffen.

Schlüsselwörter: Objekterkennung, Faltungen neuronale Netzwerke, YOLO, Raspberry Pi, Parkplatzanalyse, Computer Vision.

Table of contents

I. List of abbreviations

YOLO	**Y**ou **O**nly **L**ook **O**nce
AP	**A**verage **P**recision
mAP	**M**ean **A**verage **P**recision
CNNs	**C**onvolutional **N**eural **N**etworks
DCNNs	**D**eep **C**onvolutional **N**eural **N**etworks
DHBW	**D**ual **H**ochschule **B**aden-**W**ürttemberg
IoU	**I**nterception **o**ver **U**nion
TFlite	**T**ensor **F**low Lite

II. List of tables

III. List of figures

1. Introduction

Nestled in front of the Dual Hochschule Baden-Württemberg building in Stuttgart is an open-air bicycle parking lot with 288 spaces, serving students and employees. Atop the university structure, a camera captures the street, focusing on the bicycle parking lot and streaming continuous video sequences via a Raspberry Pi.

The objective of this project is to develop an advanced object detection system capable of analyzing these video streams. The primary goal is to accurately identify and categorize bicycles and motorcycles utilizing the parking lot. This effort serves as a precursor to the implementation of a comprehensive statistical analysis system that evaluates parking utilization patterns across various weather conditions and throughout the year.

To attain this goal, the project explores deep convolutional neural network models designed for object recognition in images and videos. An evaluation of various existing models is conducted, to pinpoint the ones that could align seamlessly with the specific requirements and available computing resources. Emphasis is placed on contemporary models, considering the rapid evolution of machine learning and computer vision technologies.

"Given the unique focus of the application on specific objects, bicycles and motorcycles, it is necessary to customize the models by training them with a specific image dataset. A suitable, freely available training set is selected, and the training phase is executed with finely tuned parameters for reliable object detection. Subsequently, the analysis of training results is followed by tests on a recorded video capturing a typical day at the parking lot, condensed into a one-minute clip. This approach allows for the evaluation of models in a real scenario, considering factors such as varying lighting conditions and object occlusions. Insights gained from these real-world tests inform the selection of an optimal model for the intended purpose.

This project unfolds in two distinct phases. In the initial phase, a comprehensive analysis led to the selection of two YOLOv5 models. This involved careful consideration of model performance, dataset suitability, and training outcomes. The

chosen models will be further developed in the subsequent phase, where optimization strategies, including post-detection methodologies such as tracking, implementation of filters and fine-tuning of the confidence threshold, will be explored.

The ultimate objective is to establish a robust detection system that lays the foundation for a tool capable of analyzing parking lot usage. The envisioned platform will deliver a graphical representation of the extracted data, unraveling valuable insights into the dynamics of the bicycle parking lot.

2. Literature Review

In the context of this project, it is essential to discuss Convolutional Neural Networks (CNNs) and the use of them for object detection, particularly in the identification of such specific objects as bicycles and motorcycles in a parking area.

2.1. Convolutional Neural Networks for object detection

Convolutional Neural Networks (CNNs) are a type of deep learning algorithm specifically used for tasks that involve the processing of pixel data. They are well-suited for tasks such as object detection, image classification, and image segmentation.

Object detection, as a computer vision solution, involves identifying objects and their locations in an image. An object detection system, such as the one used in our project, returns a bounding box that describes the coordinates of the objects in an image, and a confidence level score indicating the system's certainty about the accuracy of the prediction[1]. An example of an object detection task performed on an image is shown in figure 1.

Figure 1. Example of Object detection with Bounding box and score in percentage. (Eden AI. (n.d.))

[1] Gallagher, J. (2023).

Models like YOLO, MobileNet or EfficientDet, are implementations of deep convolutional neural networks but have unique architectural features and innovations to address specific challenges. These models are developed by researchers and organizations to advance the field of computer vision and deep learning and simplify the use of CNNs to accurately identify and locate objects.

But to make accurate predictions in real-world scenarios, it is crucial to train the model on custom datasets that contain labeled Images of the desire instances to detect. It provides the necessary examples for the model to learn and generalize patterns, textures, shapes and features specific to the target objects.[2] The trained model is already able to predict bounding boxes around detected objects and assign class labels to these boxes, these detections can be optimized with post processing techniques to enhance accuracy and keep only the most confident detections.

2.2. Object Detection accuracy

Object detection accuracy is a critical measure of how well an object detection model performs in identifying and localizing objects within images or videos. To know the performance of the model we need to evaluate if the predicted class is the correct class and how close the predicted bounding box is to the ground truth. The accuracy of an object detection model is typically evaluated using various metrics to assess its performance. These metrics include precision, recall, mean average precision (mAP), and intersection over union (IoU), among others[3]

The metrics mentioned and the relationship between them are shown in the figure 2. We can observe that the Mean average Precision evaluates the global performance of the object detection model, but to truly understand its meaning lets first look to the metrics related to it.

[2] Ultralytics. (N.D. a.).
[3] Zenggyu. (2018)

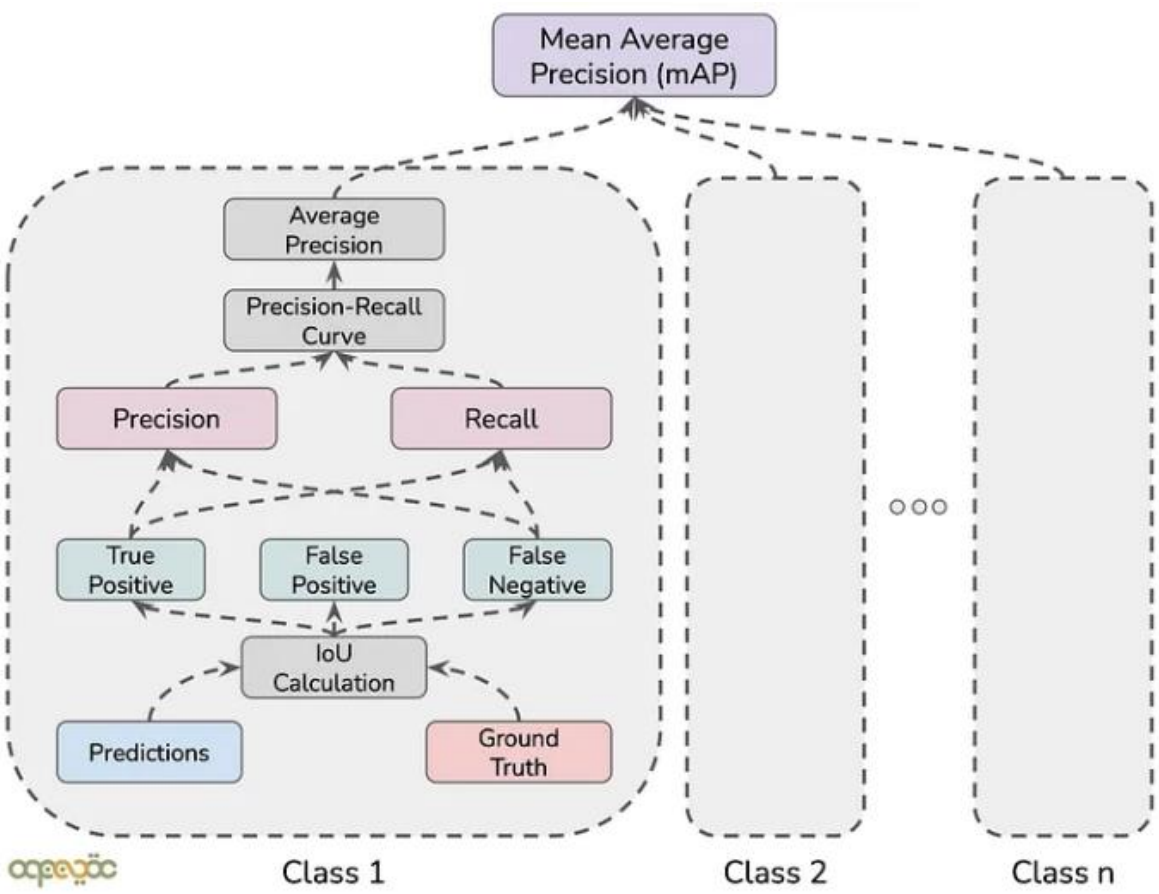

Figure 2. Metrics to evaluate Object detection. (Anwar, A. 2022)

2.2.1. Intersection over Union (IoU)

The IoU measures the overlap between the predicted bounding box, coming from the model detection, and the ground truth bounding boxes, what were manually annotated in the image.[4] The figure 3 give a graphic explanation of how the intersection over union is calculated. Higher values of IoU describe a more correct localization of the object in comparation with reality.

[4] Rezatofighi (2019)

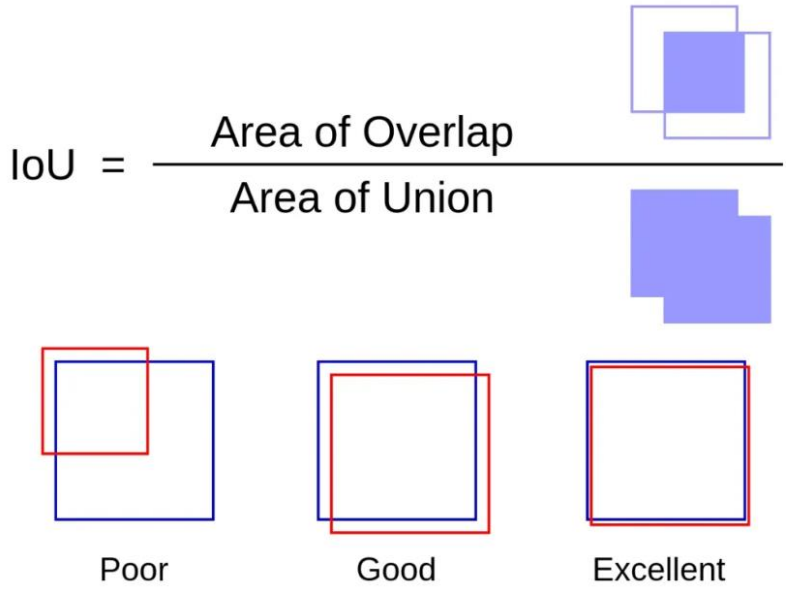

Figure 3. Calculation of the Intersection over Union Metric (Tomar, N 2023)

2.2.2. Precision, Recall and F1-Score

Precision measures the proportion of correctly identified objects among all identified objects, and is calculated using the following formula:

$$Precision = \frac{Correct\ Predictions}{Total\ Predictions} = \frac{TP}{TP + FP}$$

Where **TP** stand for **T**rue **P**ositives (correct detected objects) and **FP** for **F**alse **P**ositives (wrong detected objects).

A correctly identified object is defined as an object where the intersection over union value is higher as a specific cutoff point, that is called IoU-threshold and is usually set to 0.5.[5] If for example a predicted object has a detection overlap bigger as the stetted threshold the detection is going to be classified as True positive, and as False positive if it's below this point.

[5] Kukil, & Kukil. (2023).

Recall, also known as sensitivity, is the ratio of True Positive predictions to the sum of True positive and False Negative predictions, and is calculated with the following formula[6]:

$$Recall = \frac{Correct\ Predictions}{All\ present\ instances} = \frac{TP}{TP + FN}$$

Where **FN** stands for **F**alse **N**egatives and describe the instances or objects that where not detected at all for the model.

Both recall and precision are fundamental metrics used to asset the performance of the model, but there is a single metric called F1-Score that balances and present the harmonic mean of both:

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

In mathematics the harmonic mean is one of several kinds of average and is useful to appropriately represent the average rate. In our case the harmonic mean ensure that the F1-Score is high only when both precision and recall are high, providing a balanced assessment of the model's performance. The F1-Score or score, reflect the confidence level or probability that the detected object belongs to a certain class and is present within the associated bounding box. It is obtained as result of the neural network detection.[7]

2.2.3. Precision-Recall Curve

The PR-Curve, shown in Figure 4, describe the behavior of the precision and recall metrics for different threshold values, but what is a threshold and what is used for?

A threshold is a specific value chosen by the user that acts as a cutoff point, detections made with scores below this threshold are disregarded. It is often based on the desired balance between precision and recall in the detection task and is determined through

[6] Anwar, A. (2022)
[7] Kanstrén, T. (2023)

experimentation and with help of the PR-Curve, which allow us to have a visual confirmation of what threshold is best for our application.[8]

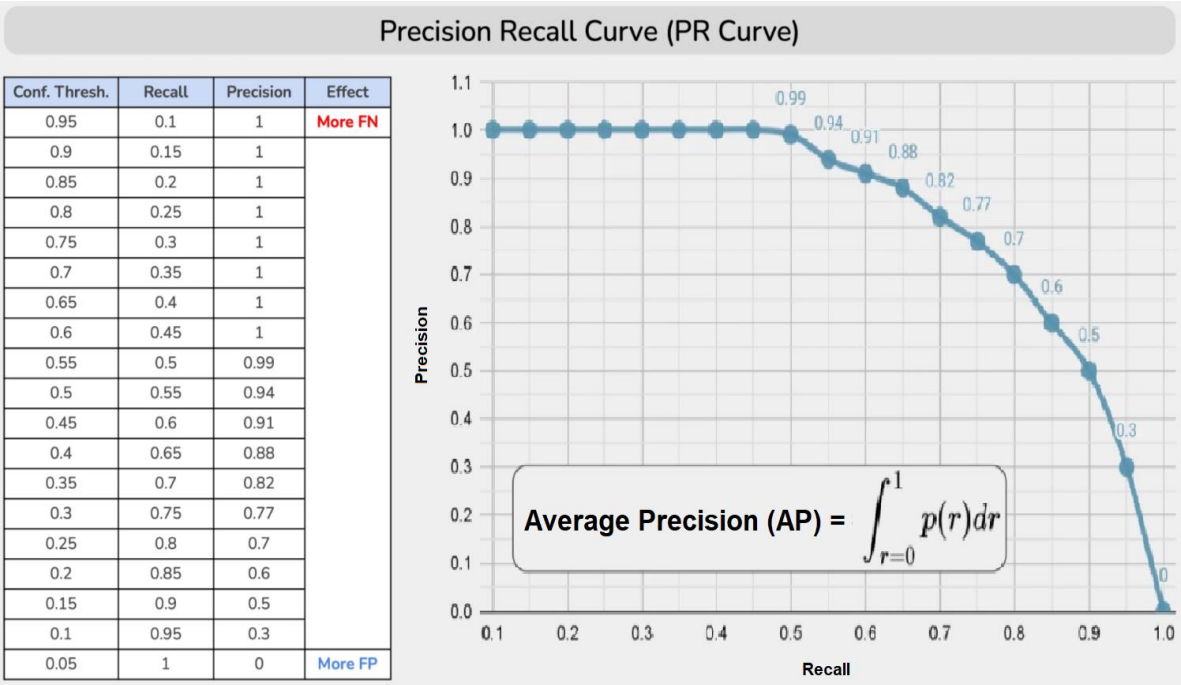

Figure 4.Simplify example of a Precision Recall Curve. (Adapted from Anwar, A. 2022)

We can look to the graph and see what happens when the threshold is set low; in this case the model will allow more detections, this helps to avoid the false negatives and therefore the recall is higher. On the other hand, setting a high Confidence threshold reduce the number of detected instances but ensures more confidence on what is being detected, hence higher Precision[9].

By observing the PR-Curve and considering the requirements, we can choose the threshold value that better fits our application. In the best scenario both Recall and Precision should be high, but in some cases is better to prioritize Precision over recall, or vice- versa, depending on the context and the applications needs, that's why experimentation also takes an important role in the probability confidence threshold determination.

[8] Anwar, A. (2022)
[9] Anwar, A. (2022)

2.2.4. Average Precision (AP)

Another useful metric derived from the PR curve is the Average Precision, which represents the area under the curve and can be calculated using integration methods such as those shown in Figure 3[10]. This metric condenses the model's performance across various thresholds into a single metric, that represents the average of all precisions achieved at different recall levels[11]. The range for AP is between 0 to 1.

AP allows comparison between different models or variations of the same model. Higher AP values indicate better performance, reflecting the model's capacity to identify relevant objects accurately across a range of recall levels. Conversely, a lower AP score suggest the model struggles to maintain a balance between Precision and recall across different thresholds, it might miss relevant detections or produce more false positives at certain operating points.[12]

2.2.5. Mean Average Precision (mAP)

If the detection task involucres more than one class, like in our case (motorcycle and bicycle classes), the AP calculation is going to be realized for each independent class. If we find the average of all AP values, we finally obtain the Mean Average Precision. While the AP focuses on a specific class, the mAP provides a more comprehensive assessment of the model performance across multiple classes. That's why this metric is a valuable measure for evaluating the overall performance of a classification model.[13]

To finish the object detection evaluation criteria lets aboard two more valuable concepts that are frequently present in model evaluations, mAP50 and mAP50-95.
As we previously mentioned the definition of True Positives depends on a stetted IoU threshold, and all the other metrics derive from there. It is valuable to think how the model will perform if we change the stablish IoU threshold. This is what the metrics indicate, they show the mAP value for different values of IoU, specifically for an IoU-

[10] AP calculation is purely a mathematical process and is not going to be more detailed in this work.
[11] Koehrsen, W. (2023).
[12] Gad, A. F. (2021)
[13] Shah, D. (2023).

threshold of 0.5 (mAP50) and for an average of multiple IoU (mAP50-95) starting at 0.5 and ending at 0.95 (usually with increments of 0.05)[14].

The mAP50 measures the accuracy of detections when considering a relative lenient overlap criterium between predicted and ground bounding box. It gives an overview of the model's performance under moderate criteria. The mAP50-95 provides a broader perspective by evaluating performance for different levels of bounding box overlap, from lenient (0.5) to strict (0.95) criteria, what gives a more comprehensive understanding of the model. In summary, higher mAP values, especially mAP50-95, signify a more robust and versatile object detection model that performs well across various levels of strictness in bounding box matching criteria[15].

[14] Kukil, & Kukil. (2022).
[15] Ultralytics. (n.d.).

3. Development of the Object detection model

Selecting an object detection model tailored to the specific requirements of our bicycle and motorcycle parking lot analysis plays a crucial role in the success of our project. To undertake this decision systematically, we identify models with potential, train them on a tailored dataset, and evaluate their performance based on both training results and real-case scenarios.

3.1. Choice of the Object Detection Models

3.1.1. Model considerations

In the process of selecting object detection models for our analysis, we have carefully considered the following models: YOLOv5, YOLOv8, SSD-MobileNet, and EfficientDet. The pre-selection of these models is grounded in key factors aligning with our project requirements.

1. Community Adoption and Documentation:
   - These models have gained widespread adoption within the computer vision community, attesting to their relevance and reliability.
   - Extensive documentation and online resources are available for these models, facilitating a seamless integration into our project workflow.

2. Compatibility with Deployment Platform:
   - The selected models are compatible with popular frameworks for edge devices, ensuring smooth deployment on the Raspberry Pi.
   - YOLOv5 and YOLOv8 feature a dedicated implementation with Ultralytics Hub, an interactive platform simplifying model training and management.
   - SSD-MobileNet and EfficientDet, implemented with TensorFlow, naturally support TensorFlow Lite, allowing straightforward deployment on Raspberry Pi. TensorFlow Lite is optimized for edge devices.

3. Balanced Trade-off Between Accuracy and Speed:
   - YOLOv5 and YOLOv8 are chosen for their commendable balance between accuracy and processing time and their potential to achieve better accuracy. A comparative study of object detection found that YOLO (yolov3 and

yolov5) provides better accuracy compared to MobileNet SSD, which provides more detection speed[16]. Additionally, research has shown that YOLOv5 has improved precision and recall in detecting objects across various scales, leading to higher mean average precision (mAP), for this reason YOLO models are anticipated to achieve superior precision and recall in object detection, capturing finer details and relationships between objects.[17]

- SSD-MobileNet and EfficientDet, designed with lightweight architectures featuring efficient convolutional layers and model structures, minimize computational load. This design makes them well-suited for deployment on devices with limited resources, such as Raspberry Pi. In addition, comparation studies have demonstrated that MobileNet-SSD offers faster detection speed compared to YOLO models, such as YOLOv3 and YOLOv5.[18] While maintaining a respectable level of accuracy, these models are likely to excel in speed.

3.1.2. Selection of models for training

In the pursuit of selecting the optimal object detection models, the conclusion has been drawn that YOLOv5 and YOLOv8 stand out as the most fitting candidates. This decision is founded on a thoughtful evaluation of various factors that align with the unique requirements of our project.

1. Emphasis on Accuracy over Speed:

YOLOv5, how already mention, is acknowledged for its potential to deliver higher accuracy in object detection. On edge devices it can operate at lower speed than the other models, but for our application accuracy is prioritized over real-time detection. This choice is driven by the need for precise results in our statistical analysis, where the thorough identification of bicycles and motorcycles is crucial. YOLOv8 is the latest release of YOLO, for this reason its results are expected to exceed its predecessor.

[16] Sabina N, Aneesa M.P, & Haseena P.V. (2022)
[17] Iyer, R., & Ringe, P. (2021).
[18] Sabina N, Aneesa M.P, & Haseena P.V. (2022)

2. Flexibility and Iterative Training with Ultralytics Hub:

The implementation of YOLOv5 and YOLOv8 through Ultralytics Hub offers a significant advantage in terms of flexibility. The interactive nature of the platform enables the training of several versions of the YOLO model with varying sizes efficiently. This flexibility allows us to iteratively experiment with different model configurations without significant time loss, ensuring that we can tailor the model to the specific requirements of our application.

3. Potential for Conversion to TensorFlow Lite:

Although YOLO models are not originally implemented in TensorFlow Lite, the possibility of conversion exists. While this process may introduce some complexity, it provides a viable solution for reducing interference speed on the Raspberry Pi. The trade-off involves carefully managing the conversion process to minimize any loss in accuracy, ensuring that the model remains effective in our application's real-world scenario.

In pursuit of a comprehensive evaluation, two specific variants of YOLO models have been selected for training and evaluation on real scenarios:

1. YOLOv5 (You Only Look Once, Version 5):
    - Testing is conducted on different variants (*YOLOv5n* and *YOLOv5n6*) to understand and evaluate its performance. These variants present diverse trade-offs in terms of computational efficiency, detection accuracy, and speed, facilitating the selection of the variant that aligns most closely with the requirements of the application.
2. YOLOv8 (You Only Look Once, Version 8):
    - Two variants, YOLOv8n and YOLOv8n_large, undergo testing. The latter is trained with a larger image size, potentially influencing its performance.

3.2. Selecting and Preparing Training Dataset:

A lot of models come already pretrained with common datasets, bringing valuable knowledge from diverse range of scenes. Leveraging this prior learning accelerates

the training process and boost performance. In line with this practice, the selected models are all pretrained on the COCO[19] dataset.

While COCO serves as a benchmark, its generic categories might not align precisely with our application's requirements. To address this, the NUImages dataset from NuScenes is utilized. The dataset, curated for autonomous driving scenarios, offers a comprehensive collection of over 93,000 images spanning more than 23 object classes. These images, closely align with our detection input, ensuring a harmonious integration into our model training pipeline. Examples of the dataset images and an image of the parling lot is shown in the figure 5.

With the aid of the NuScenes development kit, a subset of images relevant to the application was meticulously extracted and labeled, with a specific focus on the "bicycle" and "motorcycle" classes. The result is a well balance dataset that forms the bedrock of the training data. An overview of the dataset is presented in Table 3.2, illustrating key statistics and characteristics of the curated images.

Table 3.1.: Resulting dataset overview

Class	Number of images			Image Size [pixels]	Proportion [%]
	Training	Validation	total		
Bicycles	13708	3352	17060	1600	50,4
Motorcycles	13682	3097	16779	1600	49.6

To make the dataset compatible with each selected model, a preparation process is imperative. This involves formatting the dataset to match the input requirements of each model, resizing images, and organizing annotations. This meticulous preparation ensures that the models are trained on data that aligns with their specific architecture and expectations.

[19] The COCO (Common Objects in Context) dataset is a benchmark in the computer vision community, encompassing a rich set of object categories in complex scenes.

Figure 5. Examples of NuImages dataset and photo of the parking lot to analyze (third photo in the image).

3.3. Training the Models

When embarking on training object detection models, several crucial parameters need consideration. These parameters significantly influence the model's performance and its ability to generalize well to unseen data. Below are key training parameters and the rationale behind their selection:

1. Epochs:
    - The number of epochs determines how many times the entire training dataset is passed through the model. A balance must be struck to avoid overfitting (too many epochs) or underfitting (too few epochs). Typically, experimentation helps find an optimal value.[20]

2. Batch Size:
    - the batch size refers to the number of samples that are processed before the model's weights are updated. It affects the model's convergence speed and memory usage.[21] In this work, the batch size is calculated automatically with the model support methods.
3. Image Size:
    - The image size defines the resolution of the input training images. Smaller sizes can lead to faster training but may sacrifice detection accuracy, especially for smaller objects. Larger sizes enhance accuracy but demand more computational resources.[22]
4. Training Time:
    - While training time may not be a critical factor in this project, it's essential to be aware of it. Some models might take longer to converge, impacting the development cycle. For instance, while faster models may converge within a few hours, more complex architectures could require training times ranging from several hours to a day or more.

[20] Science, B. O. C. (2023)
[21] Kandel, I., & Castelli, M. (2020).
[22] Shorten, C., & Khoshgoftaar, T. M. (2019)

3.3.1. Training YOLOv8

In the exploration of object detection models, the initial training ground is YOLOv8. Specifically, the training begins with the YOLOv8n variant, utilizing the training parameters outlined in Table 3.3. The training process spans 100 epochs, a common starting point for experimentation.

Table 3.2: YOLOv8n training parameters

Epochs	Batch size	Image size
100	Auto	640

The training process involves monitoring various loss metrics, including the object loss, class loss, and box loss. Loss functions in object detection, play a pivotal role in training models. These losses quantify the disparities between predicted and ground truth values during the learning process.

The loss curves (refer to Figure 5) exhibit a typical trend, descending during the initial epochs and gradually stabilizing, suggesting effective convergence. It's crucial to note that a downward trajectory in loss metrics is desirable, indicating that the model is learning and improving its performance on the training data. Additionally, it's essential to pay attention to the divergence between training and validation losses. When validation losses deviate significantly from training losses, it's an indication that the model may be starting to overfit[23]. Throughout the training sessions, close monitoring of these trends was undertaken to ensure the model's generalization capabilities.

Signs of overfitting became apparent around the 80th epoch. In response, a strategic adjustment was implemented for the subsequent training of YOLOv8n_large, involving a reduction in the number of epochs. This adjustment aimed to optimize training time while preserving the model's generalization capabilities. This adaptive approach highlights the dynamic nature of model development, emphasizing the importance of continuous monitoring and adjustment to achieve optimal performance.

[23] Overfitting occurs when a model begins to perform exceptionally well on the training data but struggles to generalize to unseen data.

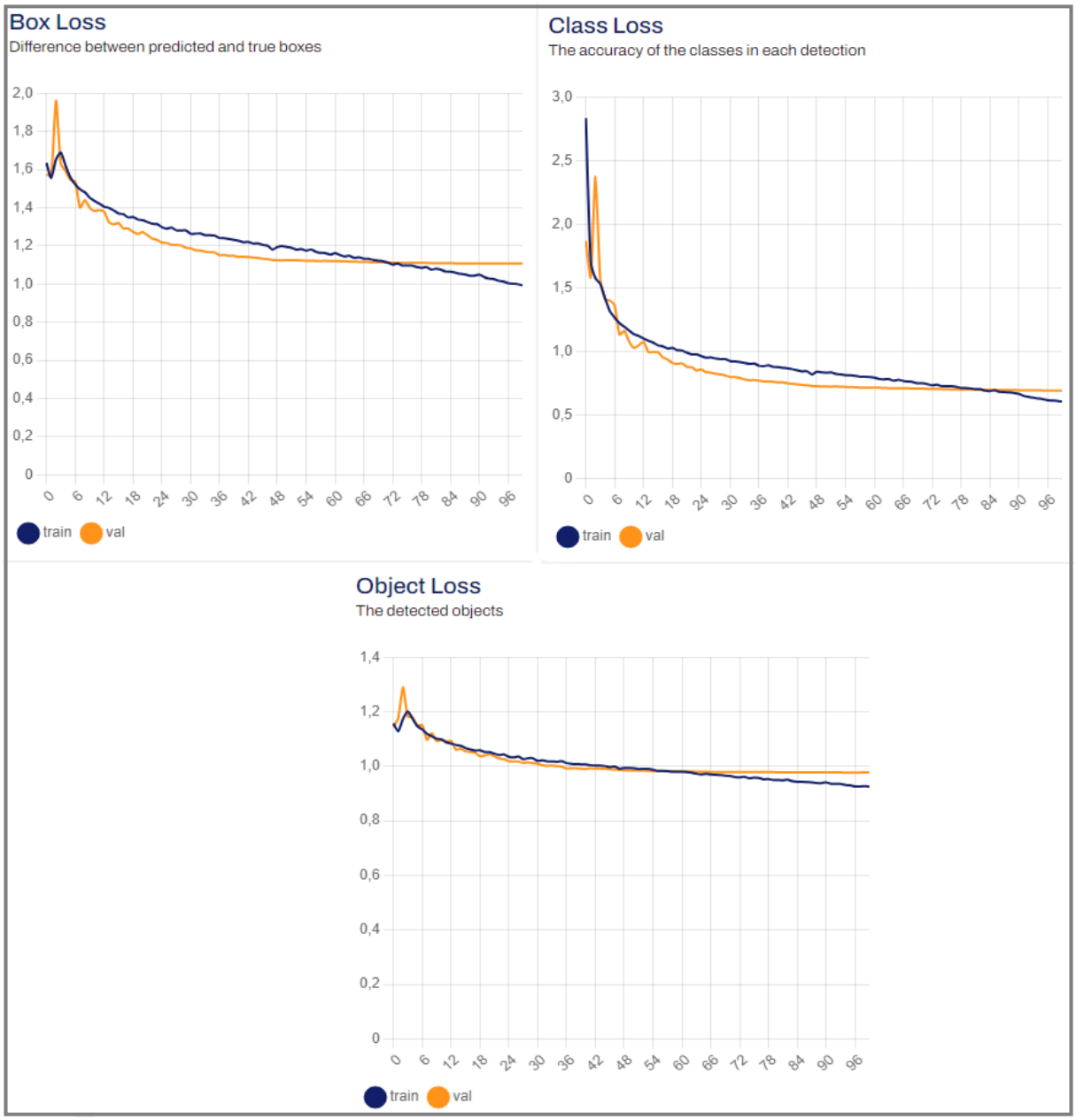

Figure 6. Box, class, and object loss curves. (Generated from Ultralytics HUB)

The parameter for the training of the YOLOv8_large are shown in Table 3.4. During the training, signs of overfitting were detected around the 60th epoch. To optimize training time and prevent excessive overfitting, the decision was made to conclude the training at this point. Notably, the training time for YOLOv8n_large was observed to be two to three times longer than that required for the smaller model.

Table 3.3.: YOLOv8n_large training parameters

Epochs	Batch size	Image size
80	Auto	1600

3.3.2. Training YOLOv5

In the training of YOLOv5, the focus is on two variants: YOLOv5n and YOLOv5n6. The key parameters for both models are summarized in the following table:

Table 3.4.: YOLOv5 training parameters

Model	Epochs	Batch size	Image size
YOLOv5n	80	Auto	640
YOLOv5n6	70	Auto	1280

These parameter choices are informed by the experience gained from YOLOv8 training, anticipating similar behavior in the YOLOv5 models. YOLOv5n6 is a version of YOLOv5n specifically adapted for optimal performance with an image size of 1280. This adaptation aligns with the goal of enhancing accuracy, particularly on small objects, without significantly increasing training and interference time.

4. Model Evaluation

4.1. Evaluation of the training results

During the training process, each model underwent an evaluation based on key performance metrics such as mAP (for different IoU thresholds) recall and accuracy. Graphical representations of these metrics show similar trends across all tested models. Figure 5 shows an extract of the YOLOv8n model curves, as an example of the evolution of the metrics during the different training epochs.

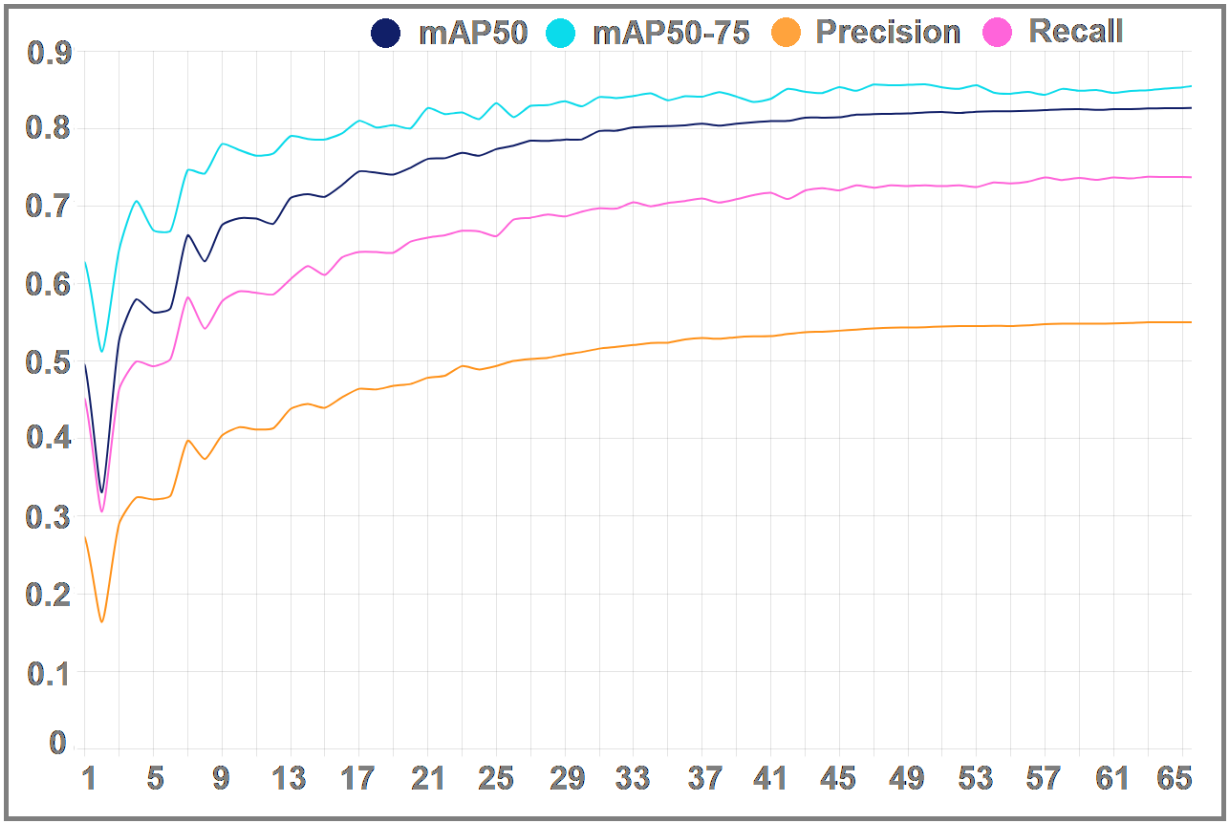

Figure 7. mAP, Precision and Recall Training curves (Generated from Ultralytics HUB)

The mAP, recall and precision curves of the four models share a distinctive pattern. In the initial epochs, typically the first three epochs, the metrics experience a drop as the model adapts to the data set. After this initial drop, the curve gradually rises, indicating an improvement in detection accuracy in the following epochs. Notably, the rise tends to slow down, suggesting that the models are approaching an asymptotic limit. Finally, the curves converge to stable values, indicating a constant level of accuracy achieved.

These consistent patterns across the mAP, recall, and precision curves showcase the models' convergence to optimal detection performance over the training epochs. The numerical representation of these trends is further summarized in Table 4.1, providing the final values for mAP50, mAP50-95, precision, and recall for each model.

Table 4.1.: Training results: mAP, Precision and Recall

Model	mAP50	mAP50-95	Precision	Recall
YOLOv8n	0.831	0.556	0.869	0.735
YOLOv8n_large	0.905	0.671	0.908	0.82
YOLOv5n	0.81	0.536	0.854	0.711
YOLOv5n6	0.895	0.65	0.885	0.819

The metrics, while exhibiting small variations, collectively demonstrate a commendable level of competence for all models in the task of object detection. In general, all models consistently achieved high mAP scores (being 1 the highest mAP value posible), indicating robust performance in detecting objects with varied levels of overlap (IoU). Precision and recall, exhibit strong and uniform performance across all models.

While YOLOv8n_large demonstrates marginally superior metrics, showcasing slightly better precision and recall, it's essential to note that all models display a commendable level of performance, suggesting their suitability for object detection tasks. The final values of the loss metrics (refer to Table 4.2) also don’t show a relevant difference across all models. This consistency, while ensuring convergence and effective learning, doesn't provide a clear basis for model differentiation. Hence, the final decision on model selection should involve testing the models in a real-world detection task.

Table 4.2.: Training results: Box, class, and object loss

Model	Box loss	Class loss	Object Loss
YOLOv8n	1.108	0.691	0.977
YOLOv8n_large	0.943	0.621	1.038
YOLOv5n	1.137	0.743	0.971
YOLOv5n6	0.987	0.657	1.01

4.2. Real-World Performance Assessment

To comprehensively evaluate the models' real-world performance, tests were conducted on a recorded video capturing a typical day at the parking lot, condensed into a one-minute clip. This video includes scenarios with varying lighting conditions, object sizes, and movement patterns, mirroring the dynamic nature of the target environment.

The evaluation process involved assessing both accuracy and inference speed, conducted on two distinct platforms: a Windows computer and the Raspberry Pi. The results, summarized in Table 4.3. The inference speed is measured in milliseconds per frame analyzed.

Table 4.3.: Interference Speed of the detection task on Windows and Raspbery

Model	Interference speed Windows (ms)	Interference speed Raspberry (ms)
YOLOv8n	75	939
YOLOv8n_large	282	4486
YOLOv5n	76	808
YOLOv5n6	206	3170

The models YOLOv8n_large and YOLOv5n6 exhibit significantly longer processing times, attributed to their increased complexity and larger parameter sizes. The computational demands of these models result in extended inference times, especially on resource-constrained devices like the Raspberry Pi.

While the inference speeds may seem comparatively slow, it's important to note that, for the application, processing a frame in 1-4 seconds provides ample time to conduct a thorough analysis of the parking lot usage. Even with the slowest model taking approximately 4.5 seconds per frame for detection, images of the parking place can be captured every 5 seconds and undergo detections. This frequency yields more than enough data for a comprehensive analysis of parking space utilization, meeting the requirements for the intended application.

Upon conducting object detection on recorded videos, a common behavior was observed across all models. While each model demonstrated the capability to detect instances of bicycles and motorcycles, consistent tracking proved challenging. Detections tended to fluctuate, with objects appearing and disappearing intermittently. This variability hindered a quantitative analysis of accuracy without the application of post-detection techniques designed to enhance tracking reliability.

Figure 8. Model YOLOv8n detection task snapshot

In qualitative observations, larger models displayed the expected capability to recognize more instances. Particularly, the YOLOv5n6 model exhibited a higher level of constancy in recognizing objects, along with a better interference speed. Due to these favorable characteristics, the decision was made to exclude the YOLOv8n_large model from further consideration. Figure 8 shows a snapshot of the detection task performed by the YOLOv5n6 model, highlighting its superior accuracy in comparison with the YOLOv8n model shown in figure 7.

The smaller models demonstrated proficiency in recognizing numerous instances, yet they lacked persistence and occasionally failed to detect intricate objects. Surprisingly, the YOLOv5n model appeared to perform better than the YOLOv8n model, counterintuitive to expectations based on the training results. This finding underscores the importance of real-world testing to uncover nuances that may not be evident in training metrics alone. In the following section, an exploration of the unexpected results

is undertaken, delving into the factors that contributed to YOLOv5's superior performance over YOLOv8 in the specific application scenario.

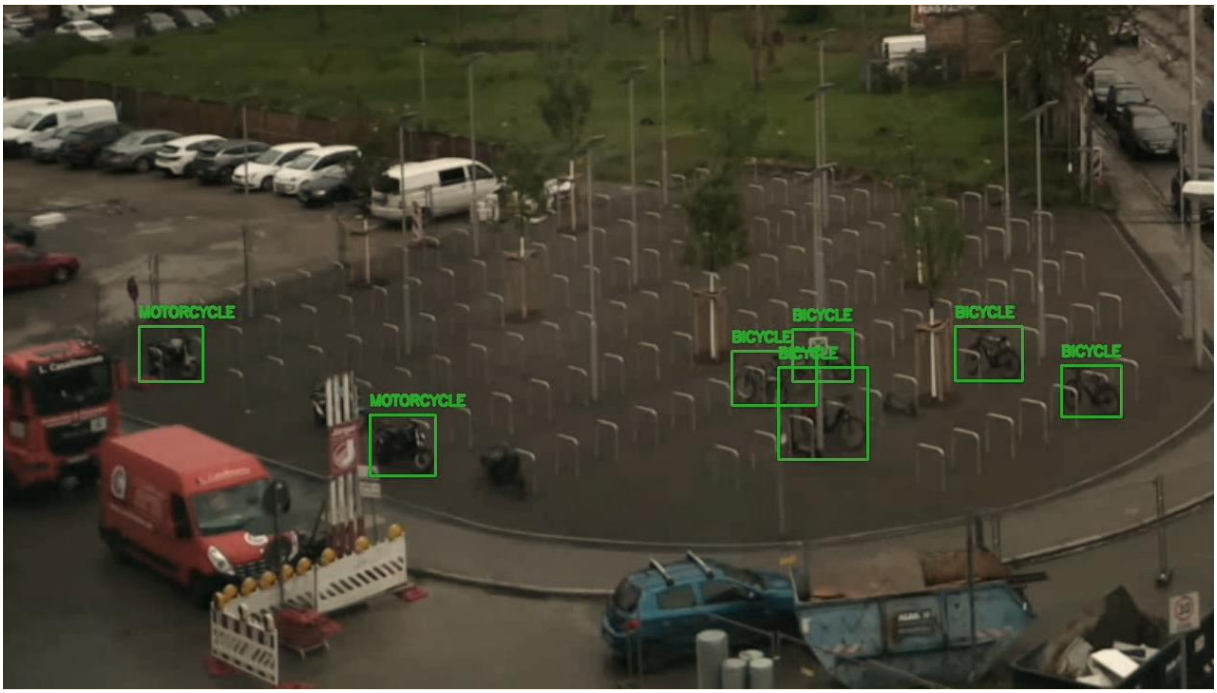

Figure 9. Model YOLOv5n6 detection task snapshot

4.3. Decoding Model Behavior: YOLOv5 vs. YOLOv8

The superiority of YOLOv5 over YOLOv8 in our specific application scenario presents an intriguing outcome. Despite both models being trained on the same dataset with similar training parameters, the observed performance contradicts the reported precision advantages of YOLOv8 in real-world applications according to existing literature[24]. Several potential factors contribute to this counterintuitive result.

One plausible explanation for the observed performance differences between YOLOv5 and YOLOv8 lies in the differences in implementation details. It's conceivable that YOLOv8 may require more fine-tuning adjustments to fully realize its optimal performance on our specific application case. Notably, post-detection techniques and threshold fine-tunning have not been implemented in the current evaluation; therefore, refining these techniques for YOLOv8 may lead to an increase in accuracy. If YOLOv8

[24] For example, the Performance Comparison of YOLOv5 and YOLOv8 Architectures for the Jurnel System Komputer. By Sary, Indri & Andromeda, Safrian & Armin, Edmund. (2023)

demonstrates better accuracy than YOLOv5 after incorporating these adjustments, it could suggest that YOLOv8 does outperform YOLOv5 but is more challenging to parametrize and adapt to a specific task. However, to make this assertion confidently, further experimentation is needed to thoroughly explore the potential of YOLOv8 in different scenarios.

Additionally, subtle disparities in hyperparameters, overlooked configurations, or sensitivity to specific settings during training and evaluation may have influenced the models' behavior. Considering the stochastic nature of deep learning models becomes another crucial aspect. Random initialization and the optimization process during training contribute to variations in model performance. Therefore, it is advisable to train multiple models, evaluate them in the specific application context, and select the one that best aligns with the project's requirements. Real-world performance may differ from expectations based solely on training results or reported benchmarks.

4.4. Model Selection

After careful consideration of inference speed and accuracy trade-offs, the decision has been made to advance the development of the YOLOv5 models. Despite the potential of YOLOv8, the current performance of YOLOv5 is already close to desirable behavior, recognizing a substantial number of instances. The optimization efforts required for YOLOv5, including post-detection techniques and threshold fine-tuning, are expected to be less compared to potential complexities with YOLOv8 models. Furthermore, the YOLOv5 trained models exhibit superior speed on the Raspberry Pi, aligning well with project goals and constraints. This, coupled with the efficiency and accuracy demonstrated by YOLOv5, makes it the preferred choice for further development.

A quantitative analysis of accuracy should be conducted to make a final decision between YOLOv5n and YOLOv5n6 models. While the YOLOv5n model exhibits faster interference, there is uncertainty about whether all instances will be detected and tracked as effectively as in the YOLOv5n6 model, especially when both models are converted to TensorFlow Lite

While real-time detection is not a strict requirement for our application, achieving lower interference times and improved computational efficiency, especially for deployment on the Raspberry Pi, is desirable. Additionally, considering that the conversion may slightly sacrifice precision for speed, it is prudent to continue developing the YOLOv5n6 model, which demonstrates superior accuracy.

5. Future Considerations and Developments

In the subsequent stages of this project, several key considerations and developments will be pursued to enhance the robustness and effectiveness of the object detection system.

5.1. Conversion to TensorFlow Lite

The conversion to TensorFlow Lite is the next step in optimizing the deployment of the selected models. However, it's essential to acknowledge that the conversion process involves various options and parameters, influencing the trade-off between inference speed and detection accuracy. These parameters, including quantization levels and optimization techniques, will be meticulously explored to achieve the most favorable balance for our specific application.

5.2. Post-Detection Techniques:

The incorporation of post-detection techniques plays a pivotal role in augmenting tracking reliability and enhancing overall detection accuracy. Techniques such as Non-Maximum Suppression (NMS) prove instrumental by filtering out redundant and overlapping detections, preserving only the most confident predictions[25]. This contributes to decluttering and refining the final detection results. Additionally, the utilization of advanced filtering methods like Kalman Filtering could also be helpful. Kalman Filtering predicts the next state of an object based on its past states, ensuring continuity in tracking, particularly when detection results display intermittent fluctuations[26]. A more in-depth analysis of these techniques will be conducted to unveil their nuanced impacts on system performance.

In conjunction with this, an in-depth analysis of threshold values must be conducted. Fine-tuning the threshold parameters allows us to adjust the sensitivity of the detection system, filtering out false positives or negatives and ensuring a more precise and reliable outcome for the analysis of the parking lot usage.

[25] Al-Badri, A. H., Ismail, N. A., Al-Dulaimi, K., Salman, G. A., & Salam, M. S. (2023)
[26] Farahi, F., & Yazdi, H. S. (2020)

5.3. Implementation Strategy:

Upon achieving reliable detections and implementing post-detection techniques, the subsequent deployment strategy involves capturing images of the parking lot at predetermined intervals. The camera, focused on the parking lot, will periodically capture frames, perform object detections, and relay the extracted data to a dedicated dashboard.

Determining the optimal frequency for capturing images is a critical consideration. The capture frequency should align with the requirements of the statistical analysis and may be influenced by the tracking capabilities or specific analysis needs. Fine-tuning this aspect will be crucial in striking the right balance between data granularity and computational efficiency.

5.4. Final Implementation and Dashboard Development:

The final phase of the project will transition towards the creation of a comprehensive dashboard to extract actionable insights from the object detection system. Key tasks include designing an intuitive interface, defining statistical metrics for in-depth analyses, ensuring seamless data integration with the Raspberry Pi-based detection model, and implementing continuous monitoring mechanisms. The goal is to establish a robust, automated system that facilitates user interaction, provides real-time analytics, and contributes valuable data for statistical analysis. Subsequent work will delve deeper into each aspect, addressing challenges related to connectivity and thorough testing before deployment in the live environment.

6. Conclusion.

In conclusion, this initial phase of the project focused on the selection, training, and evaluation of object detection models for monitoring bicycle and motorcycle usage in a university parking lot. Through a rigorous evaluation process, popular models, including YOLOv8 and YOLOv5 variants, were explored, analyzing their training results, real-world performance, and considering factors like speed, accuracy, and computational efficiency.

The YOLOv5n and YOLOv5n6 models emerged as promising candidates, demonstrating a balanced performance in terms of accuracy and interference speed. Notably, the YOLOv5n6 model exhibited superior accuracy and consistency.

Future work will involve the conversion of the chosen models to TensorFlow Lite for efficient deployment on resource-constrained devices like the Raspberry Pi. The decision to explore post-detection techniques and optimal threshold adjustments is necessitated by the observed fluctuations in detections during real-world scenarios. Addressing these fluctuations becomes crucial for ensuring the proper recognition of all instances, enhancing tracking reliability, and fine-tuning detection results.

The dataset utilized in the project played a key role in achieving robust and effective model training. The dataset's efficacy became apparent through its impact on the training behavior and resulting metric curves. The diverse and representative nature of the images in the dataset, their size, and their correspondence to the input detection frames allowed the models to generalize well to various scenarios encountered in the target environment and improve their capabilities for small object detection. This underscores the significance of a well-curated dataset in shaping the success of object detection models for specific applications.

In reflection, the project's outcomes revealed interesting insights into the performance of YOLOv5 compared to YOLOv8. While the training phase initially suggested that YOLOv8 might outperform YOLOv5, the real-world testing scenarios unveiled a

different story. YOLOv5, specifically the YOLOv5n and YOLOv5n6 variants, exhibited a surprising superiority in accuracy during practical implementations.

This unexpected success of YOLOv5 over YOLOv8 underscores the nuanced interplay between model architecture, dataset characteristics, and the intricacies of real-world detection tasks. The empirical evaluation in practical scenarios serves as a crucial benchmark for gauging a model's true effectiveness, guiding the decision-making process towards models that align optimally with the project's objectives.

The project's final goal is to deploy a robust system that captures and analyzes parking lot usage, facilitating a comprehensive understanding of patterns and behaviors. This lays the groundwork for future developments, including the creation of a dashboard for statistical analysis and real-time insights.

IX. Bibliography

Gallagher, J. (2023). *What is Object Detection? The Ultimate Guide.* Retrieved 15.11.2023 from https://blog.roboflow.com/object-detection/.

Ultralytics. (N.D. a.). *Train custom data*. GitHub. Retrieved 15.12.23 from https://github.com/ultralytics/yolov5/wiki/Train-Custom-Dat

Zenggyu. (2018). *Zenggyu的博客 - An introduction to evaluation Metrics for object Detection.* Retrieved 12.12.2023 from https://blog.zenggyu.com/posts/en/2018-12-16-an-introduction-to-evaluation-metrics-for-object-detection/index.html

Rezatofighi, Hamid and Tsoi, Nathan and Gwak, JunYoung and Sadeghian, Amir and Reid, Ian and Savarese, Silvio (2019). *Generalized Intersection over Union*. *The IEEE Conference on Computer Vision and Pattern Recognition.* Retrieved 12.12.2023 from https://giou.stanford.edu

Kukil, & Kukil. (2023). *Intersection over union IOU in object detection segmentation*. LearnOpenCV. Retrieved 15.12.23 from https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/

Anwar, A. (2022). *What is Average Precision in Object Detection & Localization Algorithms and how to calculate it?* Retrieved 03.01.24 from https://towardsdatascience.com/what-is-average-precision-in-object-detection-localization-algorithms-and-how-to-calculate-it-3f330efe697b

Kanstrén, T. (2023). *A look at precision, recall, and F1-Score - towards data science. Medium*. Retrieved 12.12.23 from https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec

Gad, A. F. (2021). *Mean Average Precision (MAP) explained | Paperspace blog. Paperspace Blog*. Retrieved 05.01.24 from https://blog.paperspace.com/mean-average-precision/

Koehrsen, W. (2023). *Precision and Recall: How to evaluate your classification model*. Retrieved 05.01.24 from https://builtin.com/data-science/precision-and-recall

Shah, D. (2023). *Mean Average Precision (MAP) explained: Everything you need to know*. Retrieved 18.12.23 from https://www.v7labs.com/blog/mean-average-precision

Kukil, & Kukil. (2022). *Mean Average precision (MAP) in object detection. LearnOpenCV*. Retrieved 15.12.24 from https://learnopencv.com/mean-average-precision-map-object-detection-model-evaluation-metric/

Ultralytics. (N.D. b.). *YOLO Performance Metrics. Ultralytics YOLOv8 Docs.* Retrieved 15.12.23 from https://docs.ultralytics.com/guides/yolo-performance-metrics/#class-wise-metrics

Tomar, N. (2023). What is Intersection over Union (IoU) in Object Detection? Image Retrieved 03.01.24 from https://idiotdeveloper.com/what-is-intersection-over-union-iou/

Science, B. O. C. (2023). *Epoch in Neural Networks | Baeldung on Computer Science. Baeldung on Computer Science*. Retrieved 07.01.24 from https://www.baeldung.com/cs/epoch-neural-networks

Kandel, I., & Castelli, M. (2020). *The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset*. Retrieved 07.01.24. from https://doi.org/10.1016/j.icte.2020.04.010

Shorten, C., & Khoshgoftaar, T. M. (2019). *A survey on Image Data Augmentation for Deep Learning. Journal of Big Data*. Retrieved 07.01.24. from https://doi.org/10.1186/s40537-019-0197-0

Sabina N, Aneesa M.P, & Haseena P.V. (2022). *Object Detection using YOLO And Mobilenet SSD: A Comparative Study*. International Journal of Engineering Research

& Technology (IJERT). Available at https://www.ijert.org/research/object-detection-using-yolo-and-mobilenet-ssd-a-comparative-study-IJERTV11IS060065.pdf.

Iyer, R., & Ringe, P. (2021). Comparison of YOLOv3, YOLOv5s and MobileNet-SSD V2 for Real-Time Mask Detection. Available at https://www.academia.edu/49690792/Comparison_of_YOLOv3_YOLOv5s_and_MobileNet_SSD_V2_for_Real_Time_Mask_Detection

Sary, Indri & Andromeda, Safrian & Armin, Edmund. (2023). *Performance Comparison of YOLOv5 and YOLOv8 Architectures in Human Detection using Aerial Images*. Ultima Computing : Jurnal Sistem Komputer. 8-13. 10.31937/sk.v15i1.3204.

Al-Badri, A. H., Ismail, N. A., Al-Dulaimi, K., Salman, G. A., & Salam, M. S. (2023). *Adaptive Non-Maximum Suppression for improving performance of Rumex detection*. Retrieved 18.01.24. from https://doi.org/10.1016/j.eswa.2023.119634

Farahi, F., & Yazdi, H. S. (2020). *Probabilistic Kalman filter for moving object tracking*. Retrieved 18.01.24. from. https://doi.org/10.1016/j.image.2019.115751

David Kriesel, 2007, *Ein kleiner Überblick über Neuronale Netze*. Available at http://www.dkriesel.com

Ultralytics. (n.d.-a). GitHub. *ultralytics/hub: Ultralytics HUB tutorials and support*. Available at https://github.com/ultralytics/hub

Ultralytics. (n.d.-b). GitHub. *ultralytics: NEW - YOLOv8 in PyTorch > ONNX > OpenVINO > CoreML > TFLite*. Available at https://github.com/ultralytics/ultralytics

X. Appendix

X.I. Detection Videos

Example videos can be accessed on YouTube with the links below. As an alternative they can be found in Nextcloud.
All videos performe during this work can be found here:
Youtube Playlist Documentation
(https://www.youtube.com/playlist?list=PLMM3KIl0jv3nJRuRbZ8qbolJzLGZT3oPxNe)
Nextxtcloud: " 2023-JG22-T31-Fahrradabstellplatz-Mejia/05-Detection Video Results”