Université de Montréal

Internship Report at BusPas Inc. 2024

Project Title: Design and develop an efficient offline computer vision system to detect site behavior at the bus stop using the SCiNe device of BusPas

par

Michell Mercedes Payano Pérez

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Internship report submitted in fulfillment of the requirements for the degree of
Master of Science (M.Sc.) in Machine Learning

March 11, 2025

Abstract

This work contains the contribution from the internship at BusPas Inc. focused on developing an AI-powered object detection system for bus stops, leveraging fisheye cameras and edge computing to enhance urban mobility. The project tackled the challenges posed by fisheye image distortions and real-world deployment constraints. Various YOLO models (versions 8 to 11) were evaluated, along with data augmentation strategies to improve object detection performance. Results showed that training on datasets resembling the deployment environment significantly improved accuracy, particularly for classes like buses and pedestrians. Additionally, a license plate detection module was implemented, integrating object detection with optical character recognition (OCR) to extract vehicle information. The study highlights the importance of dataset composition, edge processing, and privacy considerations in smart city applications.

Acknowledgments

I would like to express my deepest gratitude to my mentor, Simon Dufort-Labbé, for his invaluable guidance and unwavering support throughout this journey. Our weekly discussions were a constant source of inspiration, filled with insightful feedback, and encouragement that greatly shaped my research. His mentorship has not only refined my technical skills but also strengthened my ability to think critically and push the boundaries of innovation.

I am also incredibly grateful to my colleagues at BusPas Inc., whose camaraderie and constant support made this experience more enriching. In particular, I want to extend a special thank you to Dr. Wissem Maazoun, CTO and Vice President of Innovation, for his leadership and dedication, which have been both motivating and inspiring. My appreciation also goes to the rest of the Computer Vision team, whose collaboration and shared expertise have contributed immensely to the success of this work.

I would also like to extend my heartfelt thanks to Aaron Courville, my academic supervisor, for his guidance and for providing me with the opportunity to learn and grow under his supervision. Lastly, I am deeply thankful to Mila and the University of Montreal for granting me the privilege of being part of such prestigious institutions. The intellectually stimulating environment, the access to incredible resources, and the opportunity to engage with leading researchers in the field have been instrumental in shaping my academic journey.

Contents

List of tables

List of figures

List of Acronyms and Abbreviations

CEPDOF	Challenging Events for Person Detection from Overhead Fisheye images dataset
FRIDA	Fisheye Re-Identification Dataset with Annotation
LOAF	Large-Scale Person Detection and Localization using Overhead Fisheye Cameras dataset
LP	License Plate
mAP	Mean Average Precision
OCR	Optical Character Recognition
OWL	Open World Localization Model
SAM	Segment Anything Model
SORT	Simple Online and Realtime Tracking

YOLO	You Only Look Once

Chapter 1

Introduction

The rapid urbanization of cities worldwide has led to a growing demand for intelligent transportation systems that can enhance mobility, safety, and efficiency [**22**][**24**]. In response to this need, artificial intelligence (AI) has played a pivotal role in transforming urban transportation through real-time surveillance, traffic monitoring, and pedestrian detection. Object detection in urban environments is a fundamental challenge that requires robust and efficient models capable of handling diverse and complex scenarios. This report focuses on the development and deployment of AI-driven object detection solutions at bus stops, leveraging cutting-edge deep learning techniques and edge computing devices.

BusPas Inc., a Montreal-based startup founded in 2019, is at the forefront of this transformation by integrating smart city technology into public transportation. The company has developed SCiNe, an innovative smart display embedded in bus stops that utilizes IoT and AI to improve both user experience and operational efficiency. By deploying advanced computer vision models on edge devices such as the NVIDIA Jetson Orin Nano, BusPas aims to enhance urban mobility while addressing key challenges related to data privacy and network limitations. Unlike traditional cloud-based approaches, which suffer from latency and privacy concerns, edge-based inference ensures real-time processing without requiring constant data transmission to remote servers.

One of the primary challenges in object detection at bus stops is dealing with images captured from fisheye cameras, which introduce significant distortion. Standard object detection models trained on perspective images often struggle with such variations, leading to reduced performance. Therefore, one of the objectives of this project is to explore the adaptation of state-of-the-art YOLO models (versions 8 to 11) to address these challenges by incorporating specialized training datasets and data augmentation techniques. Additionally, an

ablation study is conducted to evaluate the impact of different dataset compositions and augmentation strategies on model performance.

Beyond general object detection, this project also extends into license plate recognition, which is crucial for applications such as traffic monitoring, security enforcement, and automated toll collection. A dedicated pipeline is developed to detect and recognize vehicle license plates when specific triggers are activated, ensuring efficient and accurate data collection. This system is designed to operate in real-world conditions, where factors such as varying lighting conditions, occlusions, and motion blur present additional challenges.

The goal of this report is to present a comprehensive study of AI-based object detection at bus stops, from dataset collection and model training to real-world deployment and performance evaluation. By leveraging edge computing, privacy-preserving methodologies, and advanced deep learning architectures, this work aims to contribute to the ongoing efforts in creating smarter and more sustainable urban transportation systems.

Chapter 2

Related Work

In recent years, the adoption of AI applications has significantly increased, particularly in enhancing the deployment of public services in urban environments. As cities continue to embrace smart technologies, AI-driven solutions play a crucial role in optimizing transportation systems, traffic management, and public infrastructure. This section provides a comprehensive review of the existing literature, exploring various methodologies, advancements, and challenges in applying AI for urban mobility and intelligent public service deployment.

2.1. Object detection for urban surveillance in smart cities

It is well known how object detection applications, as many other tasks of computer vision, have been integrated into smart city surveillance to enhance urban safety [**5**]. These algorithms are deployed on surveillance cameras strategically positioned across urban environments, such as busy intersections, public parks, and commercial areas, ensuring real-time data processing and seamless communication between surveillance units and the central monitoring system.

Among the examples of the many application of computer vision tasks applied to smart cities and mobility include the study of [**12**], where they proposed a computer vision roadside occupation surveillance system to monitor roadside parking activities, in order to alleviate traffic congestion due to double parking and inefficient use of spaces. The proposed pipeline utilizes solar-powered high-definition (HD) cameras mounted on lamp posts to continuously capture and transmit images to a cloud-based platform. These images underwent a data analytics process, which generated various metrics, including vehicle and object recognition, parking gap estimation, and available parking space calculations, among others. Another

work done by [**20**] explored the applications of object detection and tracking for people counting for public transport applications, specifically on rail replacement bus services, where bus services are employed to replace train services during periods of planned and unplanned line disruptions. The images used come from on-bus and off-bus cameras and various deep learning for object detection models, e.g. YOLO and R-CNN, and tracking algorithms, e.g. SORT and DeepSORT, were applied to detect and track individuals across frames and count unique individuals.

Another interesting work [**23**] that also deployed an edge based object detection system with the aim of improving real time traffic monitoring and public safety. The dataset used in this case consisted of images captured from traffic surveillance cameras and the objective was to detect vehicles according to their sizes, traffic signals, pedestrians and other objects. Likewise, [**42**] provides further evidence of how deploying AI-driven surveillance solutions on the edge, can improve traffic monitoring, safety, and real-time analytics. One impressive aspect of this study is they performed high-altitude real-time inspection of unmanned aerial vehicles (UAVs), in order to detect five categories of vehicles: car, truck, excavator, pile driver and crane, which serves as evidence that even real time surveillance applications using edge devices can be performed at very high altitudes.

The studies mentioned above are among the numerous works that demonstrate the effectiveness of computer vision as a powerful tool for enhancing surveillance and object detection across various conditions, such as altitudes and lighting. Furthermore, they highlight the capability of real-time detection systems to achieve high accuracy and low latency, making them invaluable for intelligent monitoring applications. However, they did not offer evidence of how this applications can be extrapolated to fisheye distorted images, as are the ones we will used for this project.

2.2. Object detection for fisheye images

For many surveillance applications, fisheye cameras are very popular since they offer a wide field view avoiding blind spots. That is why they are commonly used in parking lots, airports, traffic intersections, retail stores, hospitals, and many other expansive places. However, these cameras create a significant distortion in the images, especially around the edges of the frame, which makes it harder to identify certain objects. Also another drawback is that most object detection techniques rely on images captured by traditional perspective cameras. Traditional object detection models, designed for perspective images, struggle with orientation variations and radial appearance distortions in fisheye images.

The literature related to the usage of fisheye datasets for object detection and other computer vision tasks, is mostly related to the application of autonomous vehicles and robotics. Examples among these research projects performed in the field includes [**26**], where they propose a 3D object detection approach for fisheye images without requiring fisheye-specific training data. The key innovation is using a cylindrical projection transformation, allowing models trained on standard rectilinear perspective images to generalize to fisheye images. Another work that reflects the usage of fisheye images is done by [**31**], where they built a dataset using the CARLA Simulator, replicating the camera configuration and calibration of the WoodScape dataset to ensure seamless integration between real and synthetic data. As a result they created the largest fisheye dataset for autonomous driving applications by the time.

Even though we just mentioned two examples related to autonomous driving, overall the literature in this area is extensive. However, this work aims to apply computer vision specifically to overhead fisheye views, and at the time of writing this report, research in this area remains sparse. As it is well known, objects in fisheye images, e.g. pedestrians, appear in different shapes or sizes, and even with different orientations, such as upright, upside-down, horizontal or diagonal, specially if they are overhead view. In [**3**] some experiments were conducted in order to address the challenges of pedestrian detection in top view fisheye images, by generating perspective view patches from fisheye images and composing them into a single composite image, allowing pre-trained object detection models to be applied without additional training. Basically, the methodology was based on dividing a fisheye image into several rectangular sub-images, making the pedestrians to appear more upright, which allowed the models to detect them easily, and then, map the pedestrian detections from the composite images to the original fisheye frame by fitting a bounding box regression model to accurately transform detected bounding boxes from perspective views to fisheye coordinates. Similarly, [**11**] designed another methodology to work around the fisheye distortion by using customizing a YOLO model to handle oriented bounding boxes. As a result, the introduction of a bounding box rotation-angle loss function that determines the prediction orientation of a person, proved to ensure a robust performance, without the need of pre-processing the images or adding data augmentation, and achieved real-time inference speeds suitable for autonomous surveillance applications.

Another interesting approached followed by [**40**], combined adversarial adaptation and supervised prototype-based adaptation into a Faster R-CNN framework to enhance pedestrian detection. Basically, the adversarial adaptation aligns global image features between domains by employing a domain discriminator trained to minimize discrepancies between the source (rectilinear images) and target domains (fisheye images). The supervised prototype-based

adaptation (SPA) refines instance-level feature alignment by leveraging prototype representations for classification and bounding box regression. More recently, in the study done by [**9**], it was developed a real time passenger counting system designed for bus stops using overhead fisheye cameras and Nvidia Jetson edge computing devices to enhance privacy and efficiency. After collecting images from the bus stops, they enriched their dataset but performing data augmentations such as horizontal flipping, rotation, adjustments in saturation, contrast, and brightness, to increase the size of the available dataset for training and add variety. After comparing the results obtained from training the models DetectNet-V2, Faster-RCNN and YOLO-v4 on these fisheye datasets, the latest one achieved the highest performance. An additional real time passenger detection system done by [**33**], focused on improving automated passenger counting, specifically for autonomous public transport vehicles. Instead of using popular object detection models, such as the versions of YOLO, they presented their own model architecture that consisted on a fully convolutional neural network (CNN), followed by a feature pyramid network with a final detection head, and introduced a loss function that takes into account the scale and the angles of the bounding boxes.

In the study conducted by [**25**], they also tried to improve object detection for fisheye cameras for traffic surveillance, by using the open source dataset called Fisheye8K, which consists of 5 classes: bus, bikes, pedestrian, car and trucks. The authors of this paper proposed a pipeline that consists first of a synthetic data generation part, to transform the daytime images into nighttime using CycleGAN-based style transfer model. For the second part of the pipeline they included a zero shot open vocabulary object detection model, in order to label the unannotated images using YOLO-World. Finally, for the object detection part, several models were trained to compared their performance, and as a result, the highest accuracy was achieved by an ensemble of 3 YOLOR-D6 models. Due to the lack of publicly available fisheye datasets, [**7**] developed a data augmentation method applied to VisDrone dataset, which is a standard perspective dataset. By applying a fisheye distortion transformation to the images and using an ensemble of different object detection models, they were able to increase the robustness and accuracy for traffic scenarios.

More interesting applications using fisheye images are related to vehicle counting at intersections to assess traffic in order to get insights of vehicle trajectories, such as it was implemented by [**1**]. The framework developed combines object detection and tracking, by employing the YOLO and SORT models, with an adaptive counting mechanism. This innovative mechanism integrates a zone based counting, where vehicles are counted based on their movement through predefined spatial zones around the intersection, and a path based

counting, which handles broken trajectories due to extreme occlusion scenarios by learning movement patterns using Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

Chapter 3

Data

3.1. Dataset for object detection

At the beginning of this project it was important to build a proper dataset to conduct experiments since, at that time, we did not have a robust dataset to train and deploy the models to work with overhead fisheye images at the bus stops. Therefore, it was essential to compile and construct a dataset using both open-source images and previously collected, labeled images from SCiNe before the start of this internship. The following is a description of the first batch of the dataset gathered to train the models to detect buses, bikes, cards, pedestrians, and trucks:

- **Fisheye Re-Identification Dataset with Annotations (FRIDA)** [4]: This dataset was developed at the Visual Information Processing (VIP) Laboratory at Boston University and published in October 2022, and is based on pedestrian only. Originally, it consists of 4 videos recorded by 3 time-synchronized ceiling-mounted fisheye cameras that have fully-overlapping fields of view; however, for our purposes we used the segment videos 2 and 3 since these are the one that consists mostly of people walking around rooms and with significant occlusions.

- **Large-Scale Person Detection and Localization using Overhead Fisheye Cameras (LOAF) dataset[43]:** The LOAF dataset is designed with several essential features that make it a valuable resource for pedestrian detection and analysis as it includes diverse data across various scenes, human poses, densities, and locations. This dataset consists of over 70 videos, containing more than 42K frame images and 448K person-detection annotations, all paired with accurate location data. The collection spans 11 indoor and 40 outdoor scenes, captured at different times of the day

under varied illumination conditions, which ensure the dataset's versatility across a wide range of surveillance scenarios.

Another important feature of this dataset is that it contains the visible faces blurred with gaussian filter, in order to safeguard personal information. Since privacy protection is an important concern at BusPas Inc., we will also take into consideration blurring and pixelation techniques later on in this project and conduct experiments on the those datasets.

- **Challenging Events for Person Detection from Overhead Fisheye images (CEPDOF) dataset[6]:** This other dataset has been developed at the VIP Laboratory at Boston University and published in April 2020. It consists of 8 videos recorded by overhead-mounted fisheye cameras in one room (a small classroom), with up to 13, where there are people walking, sitting and having lunch. For our purposes we used the video sequence recall as "high activity" were people are mainly walking in through one door and leaving through the other door.

- **FishEye8K dataset[10]:** The FishEye8K dataset is an open benchmark dataset designed for road object detection in traffic surveillance using fisheye cameras. It consists of 8,000 annotated images with 157K bounding boxes across five object classes: Pedestrian, Bike, Car, Bus, and Truck. The data was collected from 22 short videos (8–20 minutes each) recorded by 18 fisheye cameras in Hsinchu, Taiwan, under diverse traffic conditions and illumination settings.

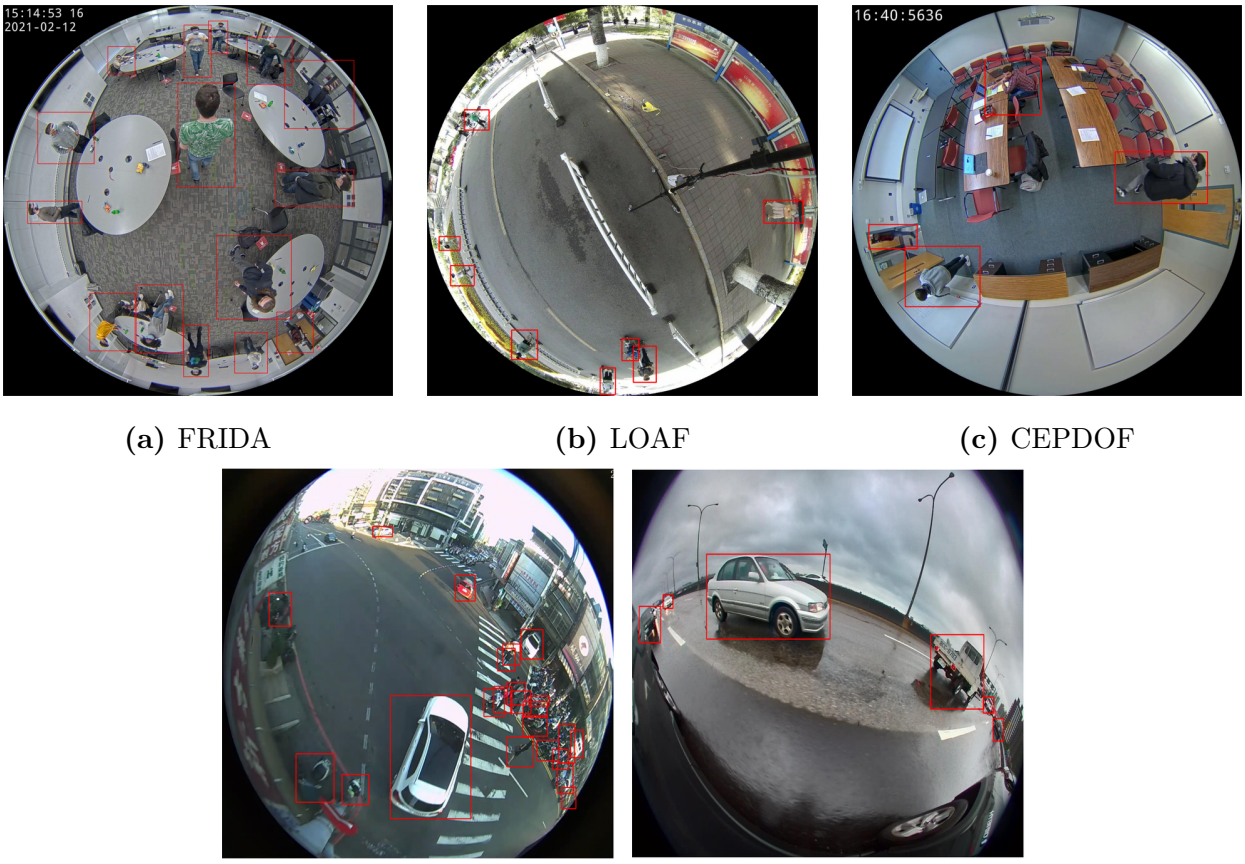
  To the best of the authors' knowledge, FishEye8K is the first publicly available dataset specifically designed for traffic surveillance with fisheye cameras. The dataset captures various traffic patterns, including urban highways and road intersections, and ensures privacy protection by blurring visible faces and license plates.

- **WoodScape dataset[24]:** This dataset was collected using an in-car fisheye dash camera; however, it was intended for self-driving scenarios, including labels for several tasks besides object detection such as semantic segmentation, visual odometry, monocular depth estimation, visual SLAM, motion segmentation, among others. Even though is not an overhead dataset, it includes labels for pedestrians, vehicles, bicycles, traffic lights and signs. However, since this one is not overhead, for our purposes, we used only use the instances related to cars and for bikes.

- **Other images:** For the rest of the images that comprise the dataset used for the first part of the experiments, some were collected using the IMX477 camera module, known for its quality and compatibility with Nvidia Jetson devices [**9**]. Also other

synthetic images [**14**] were created as a first instance to increase to have a bigger datasets that best replicated the real sites, however, in this project we will describe other methods used to create more realistic synthetic datasets using SAM [**16**] .

An example of the images for these open source datasets can be seen in **Figure 1**. Originally, some of these datasets came with rotated bounding boxes, so it was necessary to convert them to axis-aligned boxes by obtaining the four corner points of a rotated rectangle that fits a given contour or object, as shown in **Figure 2**. Furthermore, since these datasets came with different camera heights (from 8 to more than 13 feet from the ground) compared to the ones that are set in the real bus stations (8 feet), it was also necessary to remove the smallest bounding boxes. In **Figure 3** we can see the before and after the deletion process was executed on some images. As a result, our first dataset to perform the first set of experiments is highly imbalanced on pedestrians, accounting for more than 80% of the instances, as in **Figure 4**.



(a) FRIDA **(b)** LOAF **(c)** CEPDOF

(d) Fisheye8K **(e)** Woodscape

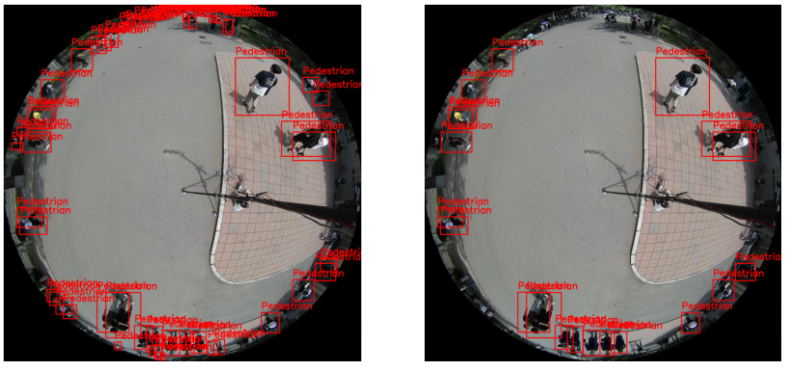
Fig. 1. Example of images from open source datasets.

(a) Original rotated bounding boxes

(b) Bounding boxes without rotation

(c) Coverted rotated bounding boxes to axis-aligned boxes

Fig. 2. Convertion of rotated bounding boxes to axis-aligned boxes.



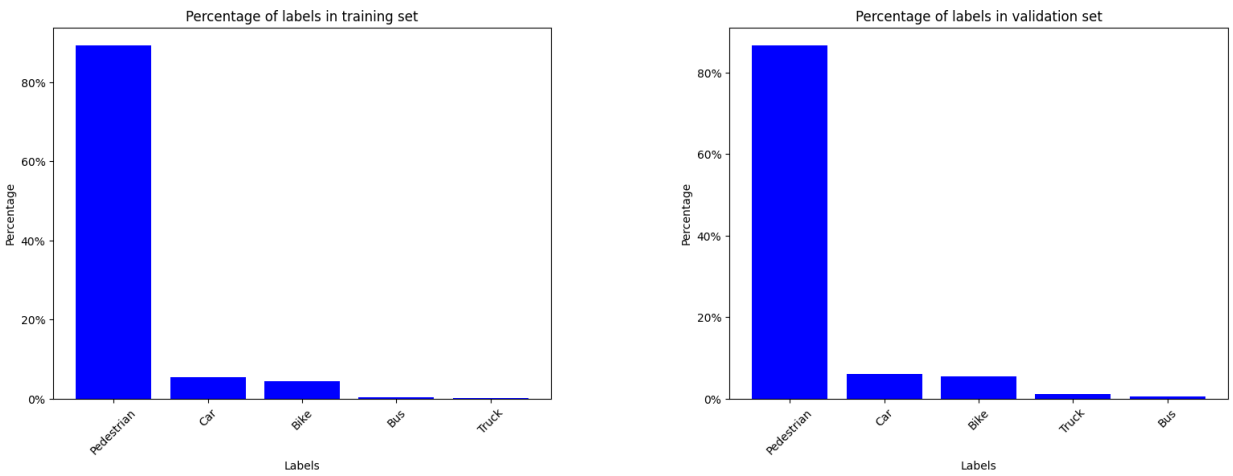
(a) LOAF dataset before removing: 65 objects

(b) LOAF dataset after removing: 20 objects

(c) Fisheye8k dataset before removing: 14 objects

(d) Fisheye8k dataset after removing: 1 object

Fig. 3. Before and after of removing smaller bounding boxes from open source datasets



(a) Training data **(b)** Validation data

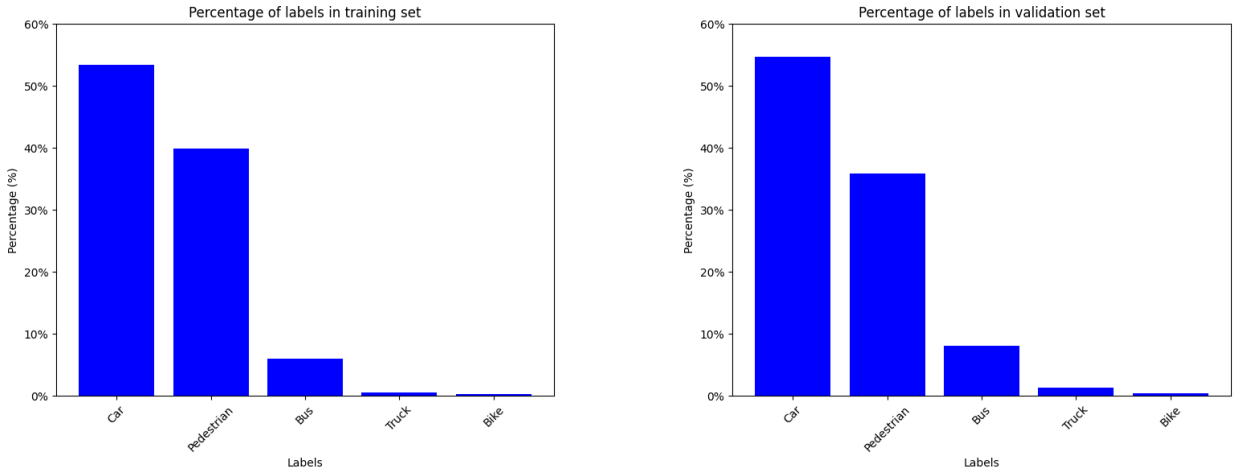
Fig. 4. Composition of the first batch dataset collected (before October 2024).

A second set of experiments were performed when more data from the SCiNe was collected around mid-October 2024. In **Figure 5** it is shown how this real site data et is collected through the fisheye camera from the smart sign which is located in several bus stops.

Fig. 5. Smart sign BusPas panel with fisheye camera and artificial intelligence edge computer NVIDIA GPU.

In order to label the new incoming data, we used Roboflow platform [**29**] to annotate it. As a result, we got a different composition from the first batch of dataset that was used to train

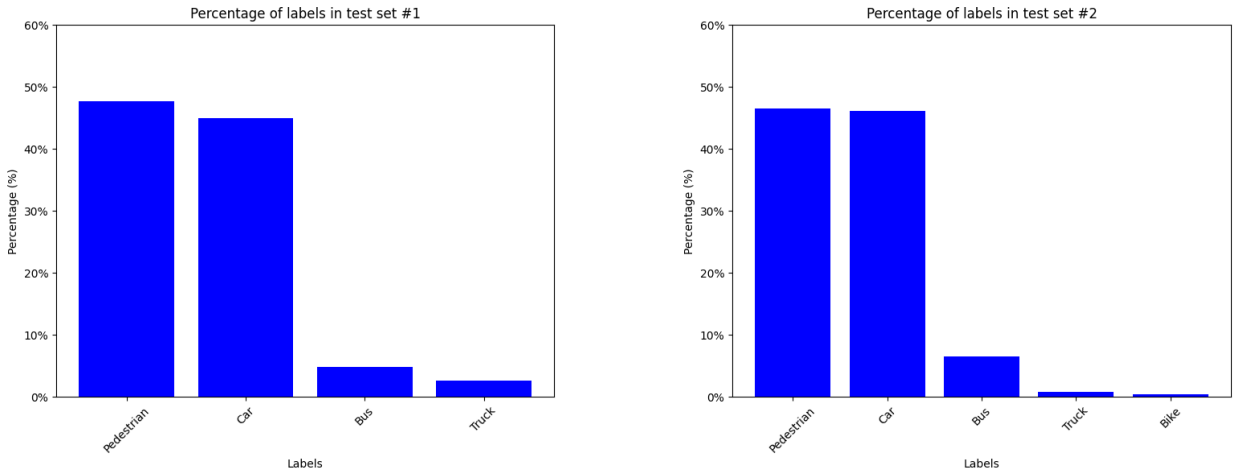
the first set of experiments. In this case, the dataset is still imbalanced, but not as much as the previous one as shows in **Figure 6**, where cars and pedestrians are the majority classes.



(a) Training data **(b)** Validation data

Fig. 6. Composition second batch dataset collected from the SCiNe device in October 2024.

The results of the models trained using different combinations of these data sets are presented in the following **Chapter 5**. For benchmarking, the primary test set consists of 200 images from a real site, but the bus stop location differs from those used in training and validation. The second test set contains 392 images, originating from the same bus stop locations as the training data.



(a) First test set (July 2024). **(b)** Second test set (October 2024).

Fig. 7. Composition of the test sets used for benchmarking.

It is crucial to note that all datasets, including those from the real site and the synthetic data derived from them, cannot be shared due to confidentiality and privacy restrictions.

3.2. Dataset for license plate detection

To perform license plate detection, it was essential to build a robust dataset. Approximately 100 videos were collected using a narrow-field lens camera located in SCiNE. The data collection period spanned from July to August 2024. Since these images are rectilinear rather than fisheye, a zero-shot, text-conditioned object detection model called Open-World Localization V2 [**21**] was used to label them.

From the compiled videos, approximately 31,918 frames were extracted and processed using OWLv2. As a result, 16,697 images were labeled, reviewed, and subsequently divided into training, validation, and test sets. Additionally, to expand the real dataset, open-source images from Roboflow [**28**] were incorporated to assess whether diversifying image sources enhances model performance.

Chapter 4

Methodology

4.1. Models

In this section, we describe the main models used in this project for either object detection and/or creation of synthetic datasets. In particular, we will focus first on describing the family of "You Only Look Once" (YOLO) models, which was originally introduced in 2016 [**27**] and has been recognized since then for achieving accurate and fast performance for real-time applications. More precisely, these kind of algorithms reframe object detection as a regression problem by mapping image pixels to bounding boxes coordinates and class probabilities.

Moreover, for YOLO models a single convolutional network predicts multiple bounding boxes and their corresponding class probabilities simultaneously, therefore it is trained on complete images and focuses directly on optimizing detection performance. Unlike sliding window or region proposal-based approaches, YOLO processes the entire image during both training and inference, inherently capturing contextual information about the objects and their surroundings.

Specifically for this project, we used the 4 latest versions (from 8^{th} to 11^{th}). Although each version introduces several modifications to increase accuracy compared to their predecessors, they unify object localization and classification tasks into a single neural network, which is one of the attributes that make them stand out from sliding window or region proposal-based approaches (such as R-CNN). This system divides the input image into a $S \times S$ grid, and if the center of a given object falls into that grid cell, then that grid cell is responsible for detecting that particular object. Moreover, each grid cell predicts B bounding boxes and their associated confidence scores which are defined as:

$$P(Object) * IOU_{pred}^{truth}$$

Where the confidence score is equal to the intersection over union (IOU) between the predicted box and any ground truth box. At the moment of making the prediction during testing, the conditional class probabilities and the individual box confidence predictions are multiply as defined below:

$$P(Class_i|Object) * P(Object) * IOU_{pred}^{truth} = P(Class_i) * IOU_{pred}^{truth}$$

Another important thing to note is that these scores encode both the probability of the class appearing in the box and how well the predicted box fits the object.

4.1.1. YOLOv8

This version of the YOLO models has a CSPNet-based backbone for efficient feature extraction, enabling multi-scale representation of input images [**44**]. For its neck, it incorporates a hybrid of the Feature Pyramid Network (FPN) and Path Aggregation Network (PANet) architecture to fuse multi-scale features effectively, improving detection across objects of various sizes. Different to earlier versions of this family of models, the 8th version introduces an anchor free approach for the head, which is the part of the model responsible for generating the final predictions. The implementation of an anchor-free bounding box prediction, simplifies the training process by eliminating the need for anchor boxes, reducing complexity and improving adaptability.

Another important thing to highlight about this model is that it incorporates a focal loss function for classification tasks in order to enhance classification by focusing on hard-to-classify objects, addressing the class imbalance issue. Furthermore, it employs mixed-precision training and other computational optimizations, which makes it suitable for deployment in resource-constrained environments.

4.1.2. YOLOv9

Deep neural networks often suffer from significant data loss during layer-by-layer feature extraction and spatial transformation, resulting in information bottlenecks and reduced performance. To address this challenge, YOLOv9 [**39**] introduces the concept of Programmable Gradient Information (PGI), which generates reliable gradients through auxiliary reversible branches. This ensures the preservation of crucial features during the transmission of data

through deep networks, enabling deep features to maintain their key characteristics for effective task execution.

Building on its predecessors, YOLOv9 integrates PGI to mitigate data loss and optimize training outcomes. Additionally, YOLOv9 incorporates the Generalized Efficient Layer Aggregation Network (GELAN), a novel lightweight architecture based on gradient path planning. GELAN effectively balances parameter count, computational complexity, accuracy, and inference speed, making it a robust and adaptable solution for a wide range of applications. Its innovative design ensures efficient parameter utilization and computational efficacy, enhancing YOLOv9's overall performance.

The combination of PGI and GELAN allows YOLOv9 to achieve superior results in object detection tasks, particularly in lightweight models. By addressing critical issues like information bottlenecks and maximizing efficiency, YOLOv9 delivers a high-performing network architecture capable of meeting the diverse demands of modern deep learning applications.

4.1.3. YOLOv10

YOLOv10 [**38**] introduces a consistent dual assignment strategy that eliminates the need for Non-Maximum Suppression (NMS) during inference. This is achieved through a dual-label assignment mechanism, where the one-to-many branch provides rich supervision during training, while the one-to-one branch enables efficient, post-processing-free inference. By removing reliance on NMS, YOLOv10 enhances both detection speed and accuracy, making it a more efficient real-time object detection model.

In addition to this optimization, YOLOv10 refines its model architecture for an improved balance between computational efficiency and detection performance. The lightweight classification head reduces overhead by replacing traditional layers with depthwise separable convolutions. The spatial-channel decoupled downsampling method improves feature extraction by minimizing information loss, while the rank-guided block design leverages compact inverted blocks (CIBs) to optimize the network's complexity. These innovations lead to a notable reduction in parameters and FLOPs, while maintaining or even surpassing the performance of models like YOLOv8, RT-DETR, and DEYO, reinforcing YOLOv10's position as a state-of-the-art object detection framework.

4.1.4. YOLOv11

At the time of writing this report, YOLOv11 is the latest iteration in the YOLO series, introducing significant architectural advancements to enhance speed, accuracy, and multi-task adaptability [**15**]. The model improves feature extraction through the C3k2 block, which

replaces previous C2f blocks by using smaller, more efficient convolutions, reducing computational cost while maintaining accuracy. It retains Spatial Pyramid Pooling - Fast (SPPF) for multi-scale detection and introduces the C2PSA (Cross Stage Partial with Spatial Attention) mechanism, which enhances spatial focus, improving detection of small and occluded objects. The neck and head modules incorporate these innovations to better aggregate and refine features, with CBS (Convolution-BatchNorm-SiLU) blocks stabilizing training and enhancing detection precision.

The **Table 1** presents a comparative analysis of different YOLO models versions (YOLOv8x, YOLOv9c, YOLOv10X, and YOLOv11l), that are going to be used in must of the experiments, based on their detection performance, model size, and computational efficiency. As we can see The YOLOv8x model has the largest parameter count (68.2M), making it the most complex, while YOLOv9c and YOLOv11l are the lighter ones. Furthermore, YOLOv8x requires the most computations (257.8B FLOPs), making it the most resource-intensive. Overall, seems to be YOLOv11l the most efficient model, with a good balance of performance and low FLOPs.

Model	**Size (pixels)**	**mAP$^{val}_{50:95}$**	**Params (M)**	**FLOPs (B)**
YOLOv8x[34]	640	53.9%	68.2	257.8
YOLOv9c[35]	640	53.0%	25.5	102.8
YOLOv10x[36]	640	54.4%	31.8	160.4
YOLOv11l[37]	640	53.4%	25.3	86.9

Table 1. Performance metrics for different YOLO versions on the COCO dataset.

All YOLO model variations used in this study were pretrained on the COCO dataset. Subsequently, these models were trained for 100 epochs with an input image resolution of 640×640 pixels. Most hyperparameters were chosen based on established literature, meaning that the default values—such as a learning rate of 0.01 and momentum of 0.937—were applied to each model. Additionally, several data augmentation techniques, including copy-paste, random erasing, and mosaic augmentation, were also incorporated to enhance model generalization.

4.1.5. Other models

4.1.5.1. SAM. The Segment Anything Model (SAM) [**16**] is a foundation model for image segmentation, developed by Meta AI Research. It generalizes across different segmentation tasks and operate in a zero-shot fashion, also follows a promptable segmentation approach,

where a user provides input prompts that can either points, boxes, masks, or text descriptions, and the model generates a valid segmentation mask.

The model architecture consists of a pre-trained Vision Transformer, ViT-H, that processes high-resolution images and produces a compressed image embedding. Then there is a prompt encoder to encode different types of prompts, as mentioned before, using CLIP-based embeddings. At end the mask decoder is a lightweight transformer-based decoder that combines the image embedding with the encoded prompt to predict segmentation masks. In this work, the SAM was used to create augmented images as it is described in **Section 4.4**.

4.1.5.2. Open-World Localization. Open-Vocabulary Learning (OWL) [**21**] model is an advanced object detection framework designed to address open-vocabulary object detection by leveraging self-training and vision-language models. This is also a vision transformer based (ViT) that uses architectures like ViT-L and ViT-G for feature extraction and object detection.

In this model, the image and text encoders are initialized from a contrastive learning model, as CLIP, then it utilizes a detector to generate pseudo-box annotations from weakly supervised web images (WebLI dataset). This model is only going to be used to annotate the license plate images described in **Section 4.5**.

4.1.5.3. Stable Diffusion. The Stable Diffusion Model [**30**] is a powerful latent diffusion model (LDM) designed for high-resolution image synthesis. It well known for significantly reducing computational requirements compared to traditional diffusion models while retaining high image quality. This kind of model consists of a two-stage generative process which is first a latent space autoencoder, that compresses images into a lower-dimensional latent space using a Variational Autoencoder (VAE). Then it follows a U-Net-based denoising network to iteratively remove noise and generate images, and by leveraging cross-attention layers, making it promptable for different inputs such as text, images, and bounding boxes.

During this internship, a version of Stable Diffusion designed for inpainting purposes [**13**] was used to remove pedestrians from real images and replace them with the background. This process was carried out at the end of the training and experimentation phases to comply with company requirements, ensuring that real data containing personal information was not retained. In **Section 4.4**, we will be able to see an example of this technique on open source images.

4.2. Metrics

The metrics used to compare and evaluate the object detection models are Precision, Recall and the Mean Average Precision (mAP). All these metrics come from the confusion matrix because the final stage of object detection models is to classify correctly the objects detected according to their bounding boxes.

- **Precision**, specifies the proportion of all the observations that the model classified as positive respect to the ones that are actually positive.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall**, also referred as True Positive Rate (TPR), specifies the proportion of the actual positives objects that were correctly classified as their true value.

$$Recall = \frac{TP}{TP + FN}$$

- **Mean Average Precision**, refers to the mean of the average precision for all the classes.

$$mAP = \frac{1}{n} \sum_{k=1}^{n} AP_k$$

Where n is the number of classes in the dataset and k is the given class. The average precision is calculated as a combination of the precision and recall:

$$AP = \sum_{f=0}^{i-1} [Recall_{(f+1)} - Recall_{(f)}] * Precision_{(f+1)}$$

In this case f is the index of the frame and i is the number of frames for an specific class.

4.3. Data Augmentation

In order to increase the size and diversity of the dataset specifically for the object detection part of this project, many data augmentation techniques were performed. Even though these techniques can be specified as hyperparameters at the YOLO model configuration, since we used Roboflow for data labeling, it also provides the capability to expand the dataset by generating augmented copies of the images through the following methods:

- **Flip**: the images are flipped both horizontally and vertically to help the model generalize better to objects that may appear with reversed orientations or up-side down.

- **Crop**: Up to 20% of each image is randomly removed in order to produce a zooming in effect.

- **Rotation**: Images were rotated 90° clockwise, counter-clockwise and upside-down, which has the effect of changing the spatial orientation. Besides these new modifications, images are also rotated randomly within a range of -15° and +15°.

- **Shear**: The images were stretched along both the horizontal and vertical axis by -10° and +10°.

- **Color modifications**: The changes applied to color and on pixel intensities include grayscale, hue adjustment (colors are shifted within a range -15° and +15°), saturation (vibrancy of colors is adjusted within a range of -25% and +25%), brightness (the intensity of the pixels is changed by a -20% and +20%), and exposure (color intensity is modified by a -20% and +15%).

- **Blur and Noise**: gaussian blur effect is applied up to a 2.5 pixels kernel size and also noise is introduced where up to 0.1% of pixels are randomly altered.

4.4. Creation crowded pedestrian images

In **Figure 9** is shown the distribution of the pedestrians in the images that belong to the first batch of the dataset created with the combination of the open source data and some of the images collected with the SCiNe before October 2024. As we can see, the majority of the images contained a few pedestrians, between 1 to 3. However, one important consideration about the models assessed in this project, is that they will be deployed and mostly used at peak hours, that means when the transit services may run more frequently, so there is expected to be an agglomeration of people in the bus stops. Therefore, one alternative explored in this project for this problem was to reproduce overcrowded images using the background of real bus stops site that were available in that given moment and the pedestrians from the open source dataset.

The idea of trying to simulate scenarios with more pedestrians is inspired by the work done in [**19**], where they present a data augmentation scheme by transferring pedestrians from other datasets into the target scene by using a variant of Generative adversarial network (GAN) and using style transfer. However, this method was not performed neither in overhead nor

fisheye images, and the authors didn't make any reference of a code implementation available to the public.

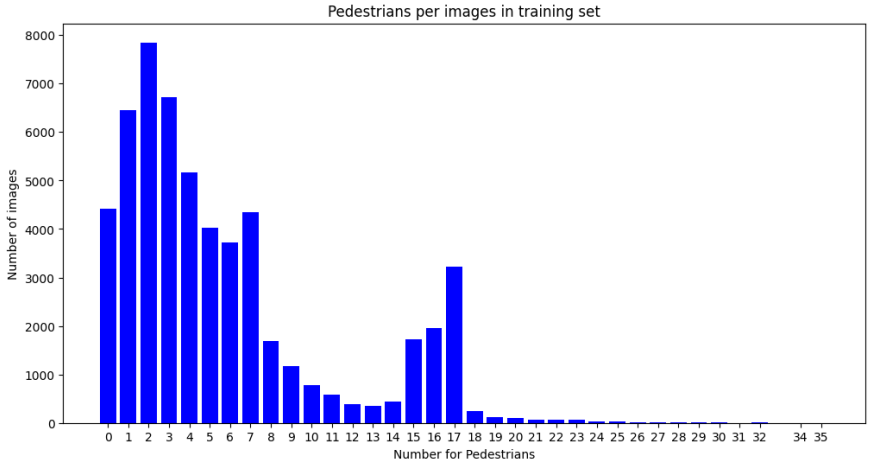


Fig. 8. Distribution of the amount of pedestrians per image for the first batch of dataset created.

Instead of using the methodology mentioned above, we decided to use SAM, which was described in **Section 4.1**. Due to privacy concerns, it is not possible to present an example from the real site after removing pedestrians with the method that will be described below, therefore in **Figure 9** we can see an example of the mask resulting operation. Basically, the strategy followed consists of the following:

(1) Identify the images of the real site that have at most one pedestrian in order to use it as a background.

(2) Identify the images from the open source data sets that have between 5 to 6 pedestrians.

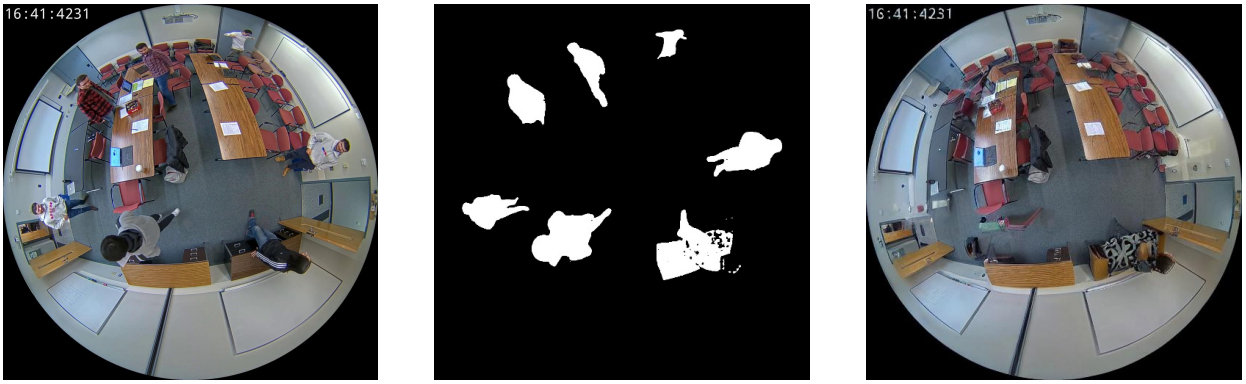
(3) Use the bounding boxes of the pedestrians identified as input for the SAM in order to generate their corresponding masks.

(4) Copy and paste each pedestrian inside of a specific region of interest from the background of real site images with no pedestrians, allowing for a certain degree of overlapping.

The effectiveness of this process, as evaluated on the test sets, will be discussed in **Chapter 5**. Additionally, due to privacy concerns, it is not possible to retain the real-site test images for an extended period. To address this, inpainting was applied to replace pedestrians with the background. While this technique was not a primary focus of the object detection

project, it represents an additional contribution by enabling the preservation of background images containing other relevant objects, such as cars and buses, which may support further experimentation.

As an alternative approach to ensure data privacy, the model was also trained using images with pixelated pedestrians, allowing an assessment of how well these models could perform in real-world scenarios while maintaining privacy standards.



(a) Original image from open source data

(b) Resulting mask of the pedestrian

(c) Resulting image after removing pedestrian

Fig. 9. Example the resulting mask of a pedestrian using SAM on an image from CEPDOF dataset and then using inpainting to delete them.

4.5. License Plate detection

In addition to focusing on the object detection aspect of fisheye data, another component of this project is the development of a comprehensive pipeline for license plate detection and optical character recognition (OCR). This involves accurately identifying license plates within detected vehicles, preprocessing the extracted plate regions, and applying OCR techniques to recognize and extract alphanumeric characters. The goal is to ensure high accuracy in real-world conditions, considering variations in lighting, angle distortions, and occlusions.

When a car is detected at the bus lane from our object detection model trained on fisheye images, a second standard camera is going to be triggered and it will start recording a 10 seconds video. Then the respective video is going to be sent to the cloud and used as input for a pipeline for license plate extraction. The pipeline for the license plate detection part consists of the following:

(1) Use of the pretrained largest version of YOLO 11 (v11X) to detect the car in the sequence of frames. As the recorded video comes from a rectilinear lens, instead of training a new model, we could use a pretrained one on COCO dataset.

(2) While detecting the car in each frame, the DeepSort [**41**] algorithm is employed to keep track of the same car and maintain consistency across frames,

(3) Use a trained object detection model on license plates using the dataset described in **Section 3.2**,

(4) Once the license plate is detected and after performing some image preprocessing, apply OCR to read the text on the license plate. It is worth noting that after an extensive search for publicly available, open-source labeled datasets for license plates from Quebec, yielded no suitable results, we opted to use pretrained OCR models for further processing and evaluation.

Since the crops of the license plate images are taken from videos where the cars are constantly moving, it is important to perform some preprocessing methods to reduce the motion blur, and this is particularly important as some pretrained OCR methods require these procedures. Among the preprocessing steps tested, they include converting the images to grayscale, then applying histogram equalization to enhance the contrast adaptively in order to help improving visibility specially in low-contrast areas, also applying non-local means denoising algorithm (NMD) to reduce as much noise as possible. Subsequently, a sharpening kernel (a 3x3 Laplacian filter) was tested to see if it could enhance edges and fine details, followed by a morphological transformation, which consists of erosion and dilation. Many of the combination of this pre-processing step were used while using the following OCRs:

- **EasyOCR**[**2**]: which is a general OCR that uses ResNet and VGG as feature extractors, LSTM for sequence labeling and CTC for decoding.

- **TrOCR**[**18**]: which is an OCR model that eliminates the need for traditional CNN and RNN components by utilizing a pure Transformer-based architecture. It consists of an image transformer as encoder (initialized from the weights of BEiT), and a text transformer as decoder (initialized from the weights of RoBERTa).

- **European & Global OCR**[**8**]: this is a simpler and light weight CNN-based OCR model designed specifically for license plate recognition. Each one consists of a deep convolutional feature extractor followed by a classification head, which can be either fully connected or convolution-based. One model was trained with either license

plates from 40 european countries and the other with worldwide license plates from around 65 countries.

Chapter 5

Results

5.1. Results of main experiments: first batch of collected data

For the models trained with the first batch of collected images, we can analyze the results presented in **Table 2**, which includes performance metrics for the validation set as well as two test sets. The table highlights the effectiveness of different YOLO models trained on two datasets, showcasing their precision, recall, and mean average precision (mAP) scores at different IoU thresholds. For a more detailed description of the datasets please refer **Table 11** at the Appendix section.

The highest $mAP_{0.5}$ on Test 1 is achieved by the YOLOv9c model (0.503) trained on "Dataset 2", which outperforms the other models in this specific evaluation metric. This dataset comprises a combination of real-world images and synthetic images generated through the data augmentation strategies described in **Section 4.3**. Additionally, another augmentation technique was employed, incorporating masks of cars and buses from real bus stop sites to enhance the representation of buses in the dataset[1]. However, by looking at the models' performance on both Test set 1 and 2, we can see that any the models is able to generalize well.

		Validation				**Test 1**				**Test 2**			
Dataset	**Model**	Precision	Recall	$mAP_{0.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$
Dataset 1	YOLOv11l	0.837	0.632	0.724	0.532	0.137	0.182	0.17	0.101	0.291	0.168	0.187	0.0964
	YOLOv10x	0.829	0.651	0.733	0.526	0.147	0.12	0.146	0.0806	0.131	0.119	0.108	0.0609
	YOLOv9c	0.83	0.646	0.737	0.543	0.164	0.358	0.27	0.153	0.356	0.214	0.204	0.107

[1]This data augmentation experiment was conducted by another member of the Computer Vision Team, where the masks, by using SAM, of buses and cars from the same real site related to Test set 1 where used to create new synthetic images.

Table 2 (continued)

Dataset	Model	Validation				Test set 1				Test set 2			
		Precision	Recall	$mAP_{0.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$
	YOLOv8x	0.86	0.607	0.724	0.539	0.13	0.176	0.146	0.0662	0.129	0.108	0.121	0.064
Dataset 2	YOLOv11l	0.946	0.944	0.974	0.837	0.839	0.394	0.484	0.316	0.39	0.227	0.249	0.142
	YOLOv9c	0.947	0.939	0.973	0.832	0.825	0.394	0.503	0.336	0.37	0.286	0.26	0.151
	YOLOv8x	0.957	0.951	0.978	0.849	0.804	0.374	0.456	0.293	0.321	0.16	0.179	0.103

Table 2. Results of different YOLO models trained with the first batch of datasets collected

A more granular analysis of the model's performance per class, as presented in **Table 3**, reveals notable variations across different object categories in Test Set 1. The model demonstrates the highest $mAP_{0.5}$ scores for cars (0.784) and pedestrians (0.66), indicating relatively strong detection capabilities for these classes. This can be attributed to the fact that these categories are generally well-represented in training datasets, leading to improved generalization. However, the augmentation of pedestrian images from open-source datasets, intended to simulate crowded environments, did not yield any noticeable improvement in pedestrian detection performance. In contrast, the augmentation of cars and buses had a positive impact, leading to increased accuracy for these classes. This is evident when comparing the performance of "Dataset 2" and "Dataset 1", where the former benefits from synthetic images, while the latter lacks these enhancements.

However, the performance for buses (0.567) and trucks (0.0) highlights potential limitations in the dataset composition. The low recall and mAP scores suggest that the model struggles with underrepresented categories, which may be due to an insufficient number of training examples capturing the full variability of real-world scenarios. This limitation is particularly evident in cases where the training data is predominantly sourced from open-source datasets, which may not fully reflect the complexities of real-world environments. Further evidence of this discrepancy is explored in the next section.

5.2. Results of main experiments: second batch of collected data

For the models trained with the second batch of collected images, the results presented in **Table 4** provide insights into the impact of incorporating newly collected real-site data, combined with various data augmentation techniques described in **Section 4.3**, as well as open-source images in some cases. It is also important to mention, that after evaluating the

Dataset	Classes	Validation				Test 1				Test 2			
		Precision	Recall	$mAP_{0.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$
Dataset 2	All	0.947	0.939	0.973	0.832	0.825	0.394	0.503	0.336	0.37	0.286	0.26	0.151
	Bus	0.991	0.994	0.995	0.992	0.65	0.5	0.567	0.52	0.428	0.39	0.308	0.212
	Bike	0.874	0.857	0.944	0.648					0.0128	0.2	0.027	0.00522
	Car	0.985	0.981	0.994	0.963	0.792	0.623	0.784	0.547	0.608	0.322	0.421	0.275
	Pedestrian	0.974	0.938	0.981	0.792	0.859	0.455	0.66	0.278	0.736	0.404	0.528	0.256
	Truck	0.913	0.927	0.954	0.766	1	0	0	0	0.0644	0.111	0.0151	0.00756

Table 3. Class metrics for the model with highest $mAP_{0.5}$ for YOLOv9c on first batch of data.

performance of the experiments performed above, we decided to take the class Trucks and merge it with Cars, therefore many of the experiments below contain 4 classes.

Overall, most models exhibit notable performance improvements on both Test Set 1 and Test Set 2 compared to those trained on the first batch of data. This suggests that the additional training samples, particularly those that better reflect real-world conditions, contributed to enhanced generalization. It is also important to highlight that Test Set 2 is derived from a data distribution similar to that of the second batch, which may further explain the observed improvements.

The highest $mAP_{0.5}$ on Test 1 (0.749) is achieved by the model YOLOv11l trained on "Dataset 5" and "Dataset 8", while on Test set 2 (0.842) is achieved by the model YOLOv9c trained on "Dataset 6". The last model also achieves a good balance between precision (0.846) and recall (0.803), meaning that it performs well while minimizing false positives and false negatives. If we look at this same model performance in Test 1, it has a $mAP_{0.5}$ of 0.739 and a precision and recall of 0.782 and 0.702 respectively, which illustrates how training on a different distribution (images from a different bus shelter) can harm model's generalization.

The highest $mAP_{0.5}$ on Test Set 1 (0.749) is achieved by the YOLOv11l models trained on "Dataset 5" and "Dataset 8", while the best performance on Test Set 2 (0.842) is achieved by the YOLOv9c model trained on "Dataset 6". Notably, the YOLOv9c model also maintains a strong balance between precision (0.846) and recall (0.803), indicating that it effectively detects objects while minimizing both false positives and false negatives. However, when evaluating the same model on Test Set 1, its $mAP_{0.5}$ drops to 0.739, with precision of 0.782 and recall of 0.702. This decline in performance highlights the challenges of generalization, as training on images from a different distribution (e.g., a different bus shelter) can negatively impact the model's ability to adapt to unseen environments.

Dataset	Model	Validation				Test 1				Test 2			
		Precision	Recall	$mAP_{0.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$
Dataset 3	YOLOv11l	0.846	0.687	0.772	0.566	0.734	0.535	0.59	0.362	0.719	0.77	0.797	0.566
	YOLOv10x	0.851	0.69	0.784	0.585	0.583	0.532	0.534	0.331	0.713	0.744	0.771	0.558
	YOLOv9c	0.879	0.666	0.762	0.57	0.684	0.498	0.593	0.381	0.842	0.763	0.841	0.574
	YOLOv8x	0.818	0.733	0.803	0.598	0.658	0.527	0.566	0.359	0.788	0.737	0.77	0.552
Dataset 4	YOLOv11l	0.757	0.69	0.726	0.52	0.784	0.523	0.625	0.395	0.692	0.774	0.77	0.512
	YOLOv10x	0.64	0.622	0.703	0.497	0.848	0.478	0.574	0.368	0.839	0.761	0.776	0.505
	YOLOv9c	0.676	0.736	0.747	0.522	0.74	0.625	0.695	0.414	0.884	0.654	0.777	0.516
	YOLOv8x	0.912	0.685	0.777	0.533	0.7	0.612	0.633	0.353	0.647	0.764	0.732	0.5
Dataset 5	YOLOv11l	0.78	0.804	0.811	0.57	0.674	0.701	0.749	0.47	0.709	0.842	0.802	0.535
	YOLOv10x	0.685	0.749	0.761	0.539	0.857	0.6	0.696	0.458	0.743	0.833	0.815	0.529
	YOLOv9c	0.756	0.766	0.802	0.565	0.795	0.621	0.771	0.483	0.732	0.833	0.825	0.56
	YOLOv8x	0.743	0.762	0.781	0.557	0.807	0.6	0.677	0.457	0.813	0.824	0.827	0.544
Dataset 6	YOLOv11l	0.779	0.796	0.809	0.598	0.781	0.632	0.729	0.468	0.752	0.818	0.816	0.533
	YOLOv10x	0.874	0.712	0.795	0.606	0.733	0.661	0.722	0.445	0.714	0.823	0.795	0.517
	YOLOv9c	0.835	0.748	0.803	0.594	0.782	0.702	0.739	0.462	0.846	0.803	0.842	0.542
	YOLOv8x	0.787	0.754	0.776	0.602	0.683	0.695	0.712	0.473	0.727	0.705	0.761	0.524
Dataset 7	YOLOv11l	0.745	0.713	0.767	0.542	0.793	0.674	0.748	0.463	0.772	0.765	0.783	0.526
	YOLOv10x	0.649	0.707	0.724	0.521	0.777	0.625	0.695	0.441	0.699	0.845	0.801	0.534
	YOLOv9c	0.76	0.734	0.779	0.542	0.774	0.646	0.71	0.451	0.691	0.826	0.772	0.529
	YOLOv8x	0.835	0.707	0.779	0.543	0.793	0.53	0.671	0.448	0.658	0.796	0.741	0.513
Dataset 8	YOLOv11l	0.806	0.804	0.842	0.597	0.856	0.685	0.749	0.451	0.653	0.855	0.765	0.521
	YOLOv10x	0.768	0.78	0.824	0.588	0.766	0.654	0.723	0.47	0.786	0.728	0.802	0.523
	YOLOv9c	0.795	0.789	0.837	0.595	0.766	0.617	0.692	0.431	0.828	0.806	0.825	0.549
	YOLOv8x	0.789	0.798	0.836	0.598	0.886	0.592	0.756	0.482	0.74	0.835	0.802	0.531
Dataset 9	YOLOv11l	0.754	0.683	0.761	0.552	0.701	0.417	0.507	0.316	0.827	0.588	0.73	0.503
	YOLOv10x	0.713	0.684	0.745	0.554	0.701	0.453	0.476	0.307	0.803	0.568	0.675	0.468
	YOLOv9c	0.79	0.684	0.762	0.562	0.716	0.438	0.538	0.345	0.805	0.526	0.694	0.49
	YOLOv8x	0.808	0.678	0.773	0.568	0.694	0.436	0.472	0.323	0.724	0.632	0.677	0.484

Table 4. Results of different YOLO models trained with the second batch of datasets collected.

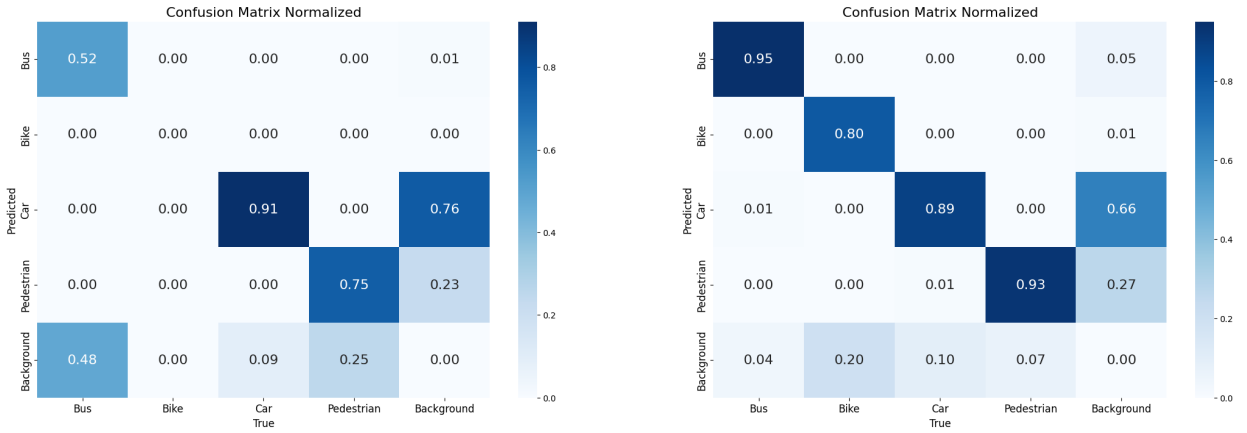
By conducting a detailed performance analysis of the YOLOv9c model across different object classes, we observe that in Test Set 2, the model achieves a $mAP_{0.5}$ above 0.9 for most classes, except for bikes, which attain a relatively lower $mAP_{0.5}$ of 0.572 (as shown in **Table 5**). Conversely, in Test Set 1, the performance is significantly lower across all classes, with the most notable drop occurring for buses, where the $mAP_{0.5}$ is only 0.591.

This discrepancy between test sets is further illustrated by the confusion matrices in **Figure 10**, where Test Set 2 shows better class separability, particularly for buses (0.95) and pedestrians (0.93), compared to Test Set 1, where the misclassification rates are notably

Dataset	Classes	Validation				Test 1				Test 2			
		Precision	Recall	$mAP_{0.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$
Dataset 6	All	0.835	0.748	0.803	0.594	0.782	0.702	0.739	0.462	0.846	0.803	0.842	0.542
	Bus	0.914	0.852	0.947	0.738	0.917	0.502	0.591	0.487	0.873	0.919	0.946	0.719
	Bike	0.59	0.333	0.363	0.165					0.746	0.591	0.572	0.17
	Car	0.85	0.834	0.909	0.645	0.626	0.9	0.851	0.549	0.832	0.825	0.907	0.644
	Pedestrian	0.984	0.971	0.992	0.826	0.803	0.704	0.773	0.352	0.931	0.878	0.943	0.634

Table 5. Class metrics for the model with highest $mAP_{0.5}$ for YOLOv9c on second batch of data.

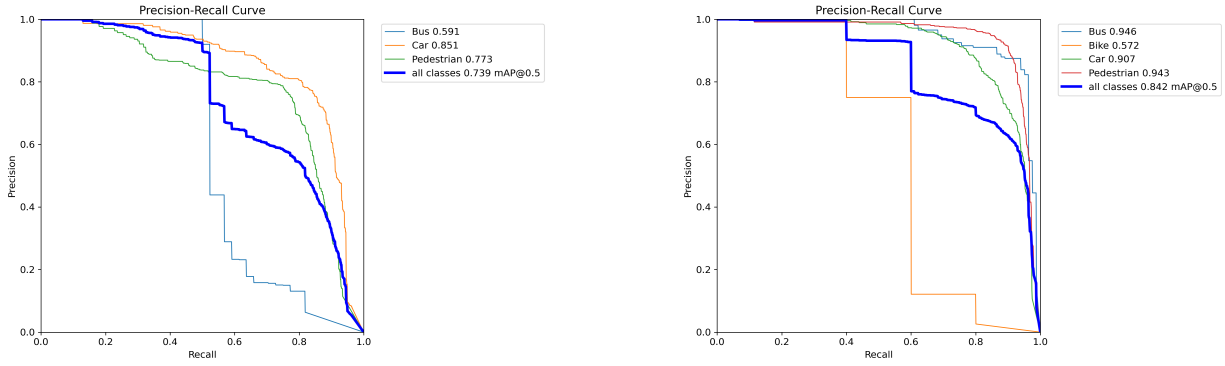
higher. For instance, the confusion matrix for Test Set 1 shows a substantial proportion of bus instances being misclassified as background, indicating a failure in distinguishing buses from their surroundings in certain conditions.



(a) Test set 1 **(b)** Test set 2

Fig. 10. Comparison of confusion matrices for both test sets

Furthermore, the precision-recall (PR) curves in **Figure 11** reinforce these findings. The curves for Test Set 2 demonstrate a consistently higher precision-recall balance across all classes (except for bikes). Meanwhile, in Test Set 1, the PR curves indicate a steeper decline in precision with increasing recall, further highlighting the challenges in generalization when applied to a different data distribution. Overall, these results suggest that the improved performance in Test Set 2 can be attributed to a better alignment between the training data distribution and the test set, whereas Test Set 1 represents a more challenging scenario, where the model struggles to generalize effectively to previously unseen conditions.



(a) Test set 1 **(b)** Test set 2

Fig. 11. Comparison of precision-recall curves for both test sets

5.3. Ablation study for object detection

In this section, we define "Dataset 4" as the Base dataset, as it exclusively consists of original images collected from real sites without any data augmentation. The experiments conducted with this Base dataset incorporate synthetic images, which were generated by strategically copy-pasting buses and cars from the same real site as Test 1 onto background images from open-source datasets and real-site bus stops[2].

For these experiments, only YOLOv9c was used, as it has consistently demonstrated strong performance across multiple prior evaluations. As observed in the results, the integration of synthetic images into the Base dataset significantly improves the model's ability to generalize across classes. This finding further reinforces the idea that a model's generalization capability strongly depends on the similarity between the training and test data distributions—i.e., the closer the new bus site (Test Set) is to the distributions present in the training data, the better the model's performance will be.

		Validation				**Test 1**				**Test 2**			
Dataset	**Model**	Precision	Recall	$mAP_{0.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$
Base (Dataset 4)	All	0.676	0.736	0.747	0.522	0.74	0.625	0.695	0.414	0.884	0.654	0.777	0.516
	Bus	0.806	0.911	0.916	0.7	0.781	0.523	0.54	0.444	0.87	0.814	0.926	0.704
	Bike	0.356	0.247	0.259	0.141					0.859	0.2	0.347	0.109
	Car	0.734	0.896	0.901	0.642	0.643	0.8	0.808	0.503	0.863	0.794	0.9	0.639
	Pedestrian	0.807	0.888	0.913	0.605	0.797	0.553	0.738	0.297	0.945	0.809	0.935	0.613
Base + Aug.1[(a)]	All	0.788	0.821	0.833	0.588	0.818	0.603	0.7	0.473	0.649	0.797	0.747	0.509
	Bus	0.869	0.932	0.965	0.771	0.923	0.523	0.576	0.516	0.83	0.951	0.943	0.718
	Bike	0.64	0.563	0.517	0.297					0.303	0.4	0.214	0.0535

[2]These data augmentation techniques were carried out by another member of the Computer Vision Team, as previously mentioned.

Table 6 (continued)

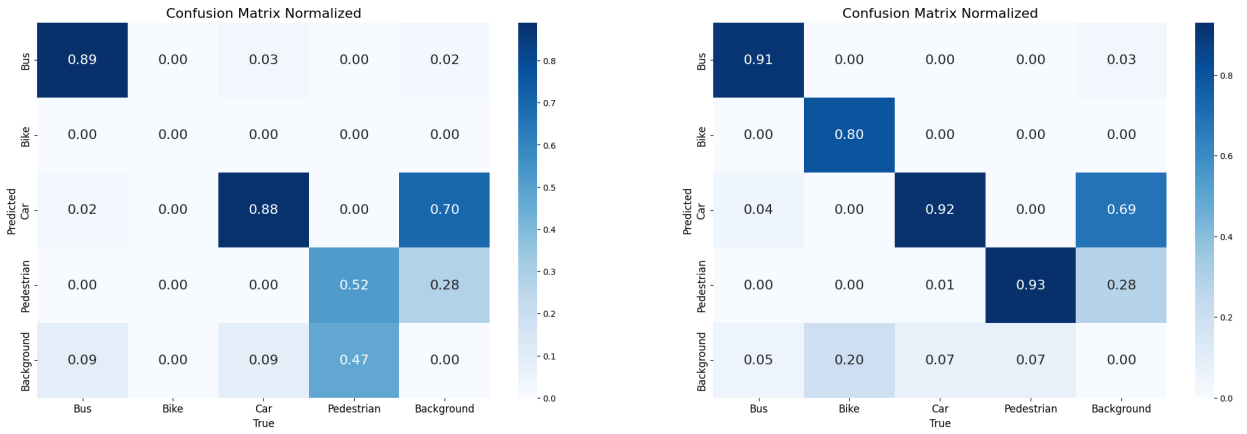
Dataset	Classes	Validation				Test set 1				Test set 2			
		Precision	Recall	$mAP_{0.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{.5}$	$mAP_{.5-.95}$
	Car	0.783	0.889	0.917	0.644	0.711	0.861	0.868	0.584	0.644	0.906	0.892	0.634
	Pedestrian	0.859	0.899	0.934	0.639	0.82	0.425	0.655	0.318	0.822	0.931	0.941	0.63
Base + Aug.2[(b)]	All	0.847	0.749	0.823	0.573	0.72	0.57	0.656	0.466	0.68	0.787	0.779	0.536
	Bus	0.917	0.888	0.955	0.768	0.664	0.545	0.633	0.562	0.829	0.946	0.937	0.732
	Bike	0.726	0.418	0.507	0.253					0.325	0.387	0.332	0.133
	Car	0.854	0.83	0.906	0.638	0.73	0.804	0.819	0.566	0.703	0.889	0.903	0.647
	Pedestrian	0.892	0.859	0.924	0.633	0.766	0.361	0.515	0.269	0.865	0.926	0.945	0.633
Base + Aug.3[(c)]	All	0.826	0.742	0.793	0.566	0.83	0.609	0.717	0.457	0.795	0.783	0.811	0.559
	Bus	0.943	0.883	0.957	0.767	0.957	0.523	0.61	0.5	0.876	0.939	0.946	0.735
	Bike	0.636	0.368	0.382	0.218					0.642	0.4	0.441	0.215
	Car	0.834	0.845	0.909	0.651	0.672	0.851	0.841	0.553	0.764	0.884	0.909	0.644
	Pedestrian	0.892	0.87	0.925	0.629	0.86	0.452	0.699	0.32	0.898	0.91	0.947	0.644
Base + Aug.4[(d)]	All	0.815	0.749	0.81	0.562	0.716	0.613	0.733	0.483	0.872	0.746	0.828	0.568
	Bus	0.922	0.903	0.963	0.771	0.667	0.545	0.737	0.572	0.887	0.864	0.948	0.74
	Bike	0.635	0.368	0.441	0.216					0.835	0.4	0.508	0.245
	Car	0.826	0.851	0.909	0.634	0.69	0.865	0.869	0.586	0.827	0.842	0.908	0.645
	Pedestrian	0.875	0.875	0.926	0.628	0.792	0.427	0.593	0.291	0.938	0.878	0.946	0.64
Base + Aug.5[(e)]	All	0.717	0.746	0.773	0.535	0.738	0.737	0.806	0.512	0.78	0.833	0.834	0.552
	Bus	0.9	0.914	0.966	0.753	0.779	0.88	0.872	0.637	0.877	0.939	0.943	0.705
	Bike	0.469	0.263	0.298	0.132					0.735	0.564	0.55	0.249
	Car	0.698	0.902	0.905	0.642	0.688	0.866	0.872	0.596	0.675	0.914	0.902	0.63
	Pedestrian	0.8	0.906	0.921	0.615	0.749	0.467	0.673	0.305	0.832	0.915	0.941	0.622

Table 6. Results of ablation study. **Notations:** [(a)] This contains a combination of the Base dataset plus augmentations on Fisheye8k dataset with the cars and buses coming from the same distribution of Test 1. [(b,c,d)] These ones contain a combination with different proportions of augmented images, also for bus and cars, using the background of real site images. [(e)] This one contains a combination of both data augmentions described above.

By analyzing the confusion matrices of both test sets in **Figure 12**, we observe clearly the class separability across must categories. This contrasts with the results presented in **Figure 10** from the previous section, where distinct class separation was evident only in Test Set 2, highlighting the improved differentiation in the current analysis.

5.4. Metadata of Object Detection

Once the object detection module in DeepStream is activated, the NvDeepSORT tracking algorithm is employed. While NVIDIA provides multiple tracking options, such as NvDCF and NvSORT, NvDeepSORT has demonstrated superior performance [**9**]. To enable the



(a) Test set 1 **(b)** Test set 2

Fig. 12. Comparison of normalized confusion matrices for both test sets for YOLOv9c in ablation study.

extraction of the metadata that results form the object detection models, a C++ script was implemented within the DeepStream modules, ensuring efficient data processing and storage. The primary objective is to generate a set of statistical insights, such as dwell time, that facilitate more informed decision-making for enhancing urban mobility. Since the SCiNE devices operate on solar power, many of them do not function continuously during the winter months or when weather conditions are unfavorable. For instance, as shown in **Table 7**, the first five rows of a dataset containing 10,496 unique tracked objects are presented. This data was collected from November 7th to 13th, 2024, at one of the real-site locations where these devices are installed.

ID	Label	FirstTime	LastTime	DwellTime(sec)	FirstLeft	FirstTop	FirstRight	FirstBottom	LastLeft	LastTop	LastRight	LastBottom
22	Car	11/7/2024 16:34	11/7/2024 16:34	2	438.28	339.72	557.62	466.38	64.90	292.69	102.01	367.57
8	Car	11/7/2024 16:32	11/7/2024 16:32	1	50.56	284.64	86.03	348.92	45.05	276.14	76.18	332.12
25	Car	11/7/2024 16:34	11/7/2024 16:34	9	507.85	331.62	572.19	424.17	109.15	355.36	226.70	479.52
11	Car	11/7/2024 16:32	11/7/2024 16:33	18	521.76	318.00	577.95	407.10	51.13	277.47	71.81	318.83
14	Car	11/7/2024 16:33	11/7/2024 16:33	1	73.25	311.04	114.05	385.35	57.86	284.11	79.05	329.49

Table 7. Metadata Table: Sample of Object Tracking Data

The metadata collected from object detection and tracking is visualized in the form of a heatmap, as shown in **Figure 13**. This heatmap represents the pedestrian count per hour for each recorded day, providing valuable insights into pedestrian activity patterns. Each row corresponds to a specific date, with the corresponding day of the week labeled, while the columns represent different hours of the day.

From the heatmap, we observe that pedestrian traffic varies significantly based on both the day of the week and the time of day. Certain peak hours emerge, particularly during late

mornings and early afternoons on weekdays, whereas weekends display a different distribution of pedestrian movement. This suggests that human mobility patterns may be influenced by factors such as work schedules, public transportation availability, or specific events in the area. By collecting data over an extended period, this analysis can be further refined to identify long-term trends, seasonal variations, and site-specific patterns. Such insights can be instrumental in urban planning, pedestrian safety improvements, and optimizing mobility solutions for different environments.

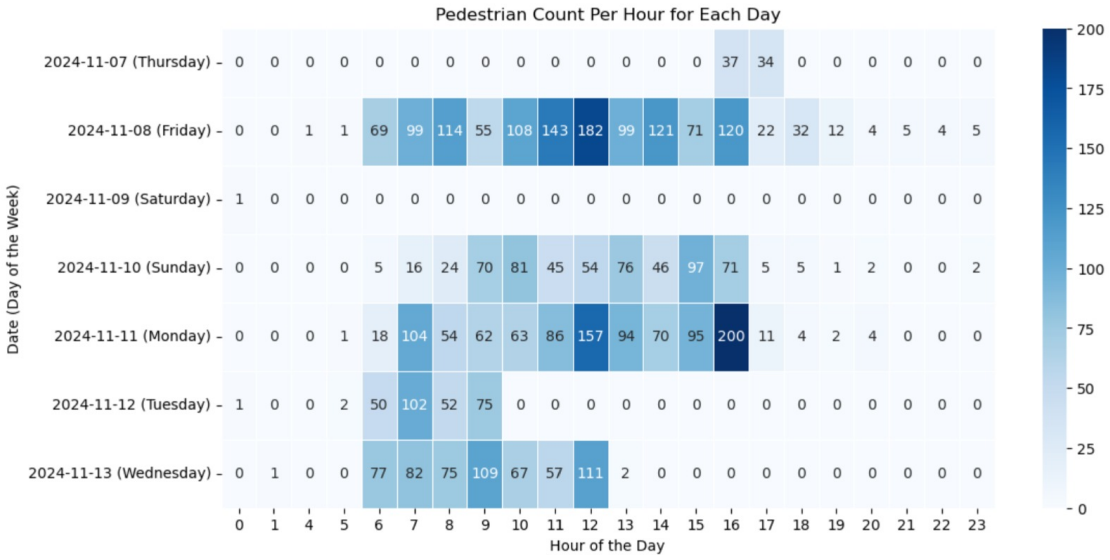


Fig. 13. Heatmap of the count of detected pedestrians per day and hour in one of the real sites.

5.5. License plate detection

After training YOLOv11x using both real-site data alone and a combination of real images with an open-source labeled dataset, the results are presented in **Table 8**. The "Real Data Only" model achieved slightly higher precision, with a score of 0.982 on the test set, compared to 0.933 for the "Real + Open Source" model. A similar trend was observed in recall values. Regarding mAP (0.5-0.95), which measures performance across various IoU thresholds, the "Real + Open Source" model performed worse than the "Real Data Only" model. Consequently, we selected the model trained exclusively on real data for integration into our automatic license plate detection and recognition system.

Once the license plate detection model was chosen, we proceeded to evaluate different OCR models, as shown in **Table 9**. The models were tested over a video of 10-second were there was only one car, where the OCR model extracted the text for each frames. The highest

Dataset	Validation				Test Set			
	Precision	Recall	$mAP_{0.5}$	$mAP_{.5-.95}$	Precision	Recall	$mAP_{0.5}$	$mAP_{.5-.95}$
Real data only	0.978	0.812	0.919	0.794	0.982	0.817	0.922	0.799
Real + Open Source	0.967	0.859	0.938	0.699	0.933	0.772	0.886	0.677

Table 8. Results for LP detection.

confidence score per sequence was recorded as the final result. While OCR Global LP had the highest numerical confidence score, a manual verification revealed that it was less accurate in character recognition compared to the Transformer-based and European OCR models. This suggests that the confidence score alone is not a fair metric to evaluate OCR performance, as global models may assign high confidence to incorrect predictions. The Transformer-based and European OCR models showed better alignment with actual ground truth characters despite having lower confidence scores.

Therefore another metric was used which is the Character Error Rate (CER). This is a simple but insightful metric that indicates the percentage of characters that are incorrectly predicted by the OCR model, so the lower the value, the closer is the prediction to the ground truth values. As a result, the OCR that achieves a higher performance is the TransformerOCR.

OCR Model	**Max. Conf. Score**	**CER**
OCR Global LP	0.9529	0.514
TransformerOCR	0.7808	0.2857
OCR Euro LP	0.6842	0.4286
EasyOCR	0.5450	0.7143

Table 9. Confidence scores and Character Error Rate of different OCR Models.

Chapter 6

Conclusion and future work

This internship at BusPas Inc. focused on improving object detection at bus stops using fisheye cameras and deploying AI models on edge devices in order to extract relevant metadata. The project aimed to enhance the accuracy and efficiency of object detection while addressing the unique challenges posed by fisheye distortions and real-world urban environments. Through extensive experimentation with various YOLO models and data augmentation techniques, the study provided valuable insights into optimizing object detection for smart city applications.

One of the key findings was that training models on datasets that closely resemble the target deployment environment significantly improves performance. Models trained with synthetic and augmented data, particularly those incorporating object from real-site images, achieved higher mAP scores compared to those trained solely on open-source datasets or from images with a different distribution. However, challenges remain. Despite improvements, object detection performance varied across different classes, specially as there persists imbalances for classes such as bikes. The impact of training data distribution on model generalization was evident, as models trained on one bus stop location struggled when evaluated on images from different sites.

The license plate detection module showed promising results, with YOLOv11x achieving high accuracy when trained on real-site data. However, the evaluation of different OCR models revealed that existing solutions might not be fully optimized for real-world conditions, particularly under low-light or motion blur scenarios. This suggests that future work should explore training custom OCR models specifically designed for this application and possibly experiment with several advanced techniques for denoising the license plates images captured in the wild, such as GANs [**17**] [**32**].

Overall, this project demonstrated the feasibility of deploying AI-driven object detection at bus stops, contributing to smart city initiatives. Future research should focus on refining model adaptation to different urban environments, exploring advanced data augmentation strategies, and further optimizing the edge computing pipeline to ensure robust and scalable real-world deployment.

References

[1] Morteza Adl, Ryan Ahmed, Carlos Vidal, et Ali Emadi, *Enhanced vehicle movement counting at intersections via a self-learning fisheye camera system*, IEEE Access (2024).

[2] Jaided AI, *Easyocr*, `https://github.com/JaidedAI/EasyOCR`, Accessed: October 2024.

[3] Sheng-Ho Chiang, Tsaipei Wang, et Yi-Fu Chen, *Efficient pedestrian detection in top-view fisheye images using compositions of perspective view patches*, Image and Vision Computing **105** (2021), 104069.

[4] Mertcan Cokbas, John Bolognino, Janusz Konrad, et Prakash Ishwar, *Frida: Fisheye re-identification dataset with annotations*, 2022 18th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2022, pp. 1–8.

[5] Kosal Dharany et Mariam Trinita, *Enhancing urban safety: The role of object detection in smart city surveillance systems*, ITEJ (Information Technology Engineering Journals) **8** (2023), no. 2, 63–72.

[6] Zhihao Duan, Ozan Tezcan, Hayato Nakamura, Prakash Ishwar, et Janusz Konrad, *Rapid: rotation-aware people detection in overhead fisheye images*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 636–637.

[7] Viet Hung Duong, Duc Quyen Nguyen, Thien Van Luong, Huan Vu, et Tien Cuong Nguyen, *Robust data augmentation and ensemble method for object detection in fisheye camera images*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 7017–7026.

[8] FastPlateOCR, `https://github.com/ankandrew/fast-plate-ocr`, 2024, Accessed: October, 2024.

[9] Pardis Ghaziamin, Kairavi Bajaj, Nizar Bouguila, et Zachary Patterson, *A privacy-preserving edge computing solution for real-time passenger counting at bus stops using overhead fisheye camera*, 2024 IEEE 18th International Conference on Semantic Computing (ICSC), IEEE, 2024, pp. 25–32.

[10] Munkhjargal Gochoo, Munkh-Erdene Otgonbold, Erkhembayar Ganbold, Jun-Wei Hsieh, Ming-Ching Chang, Ping-Yang Chen, Byambaa Dorj, Hamad Al Jassmi, Ganzorig Batnasan, Fady Alnajjar, et al., *Fisheye8k: A benchmark and dataset for fisheye camera object detection*, Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2023, pp. 5305–5313.

[11] Olfa Haggui, Hamza Bayd, Baptiste Magnier, et Arezki Aberkane, *Human detection in moving fisheye camera using an improved yolov3 framework*, 2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP), IEEE, 2021, pp. 1–6.

[12] George To Sum Ho, Yung Po Tsang, Chun Ho Wu, Wai Hung Wong, et King Lun Choy, *A computer vision-based roadside occupation surveillance system for intelligent transport in smart cities*, Sensors **19** (2019), no. 8, 1796.

[13] Huggingface, *Huggingface website*, 2024, Accessed: October 1, 2024.

[14] Rexys Inc., *Aigo.*, `https://rexys.io/`, 2023.

[15] Rahima Khanam et Muhammad Hussain, *Yolov11: An overview of the key architectural enhancements*, arXiv preprint arXiv:2410.17725 (2024).

[16] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, et Ross Girshick, *Segment anything*, Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2023, pp. 4015–4026.

[17] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, et Jiří Matas, *Deblurgan: Blind motion deblurring using conditional adversarial networks*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8183–8192.

[18] Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, et Furu Wei, *Trocr: Transformer-based optical character recognition with pre-trained models*, Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 13094–13102.

[19] Songyan Liu, Haiyun Guo, Jian-Guo Hu, Xu Zhao, Chaoyang Zhao, Tong Wang, Yousong Zhu, Jinqiao Wang, et Ming Tang, *A novel data augmentation scheme for pedestrian detection with attribute preserving gan*, Neurocomputing **401** (2020), 123–132.

[20] Chris McCarthy, Hadi Ghaderi, Felip Martí, Prem Jayaraman, et Hussein Dia, *Video-based automatic people counting for public transport: On-bus versus off-bus deployment*, Computers in Industry **164** (2025), 104195.

[21] Matthias Minderer, Alexey Gritsenko, et Neil Houlsby, *Scaling open-vocabulary object detection*, Advances in Neural Information Processing Systems **36** (2024).

[22] Pantrigo J.J. Salgado L. Montemayor, A.S., *Special issue on real-time computer vision in smart cities*, Real-Time Image Proc 10, 723–724 (2015), 2014.

[23] Sushma Nagaraj, Bhushan Muthiyan, Swetha Ravi, Virginia Menezes, Kalki Kapoor, et Hyeran Jeon, *Edge-based street object detection*, 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), IEEE, 2017, pp. 1–4.

[24] United Nations, `https://www.un.org/uk/desa/68-world-population-projected-live-urban-areas-2050-says-u` Accessed: January, 2025.

[25] Long Hoang Pham, Quoc Pham-Nam Ho, Duong Nguyen-Ngoc Tran, Tai Huu-Phuong Tran, Huy-Hung Nguyen, Duong Khac Vu, Chi Dai Tran, Ngoc Doan-Minh Huynh, Hyung-Min Jeon, Hyung-Joon Jeon, et al., *Improving object detection to fisheye cameras with open-vocabulary pseudo-label approach*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 7100–7107.

[26] Elad Plaut, Erez Ben Yaacov, et Bat El Shlomo, *3d object detection from a single fisheye image without a single fisheye training image*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3659–3667.

[27] J. Redmon, *You only look once: Unified, real-time object detection.*, Proceedings of the IEEE conference on computer vision and pattern recognition (2016).

[28] Roboflow, *Roboflow website*, 2024, Accessed: September 30, 2024.

[29] Roboflow, `https://roboiňĆow.com`, 2024, Accessed: October, 2024.

[30] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, et Björn Ommer, *High-resolution image synthesis with latent diffusion models*, Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 10684–10695.

[31] Ahmed Rida Sekkat, Yohan Dupuis, Varun Ravi Kumar, Hazem Rashed, Senthil Yogamani, Pascal Vasseur, et Paul Honeine, *Synwoodscape: Synthetic surround-view fisheye camera dataset for autonomous driving*, IEEE Robotics and Automation Letters **7** (2022), no. 3, 8502–8509.

[32] Theophil Trippe, Martin Genzel, Jan Macdonald, et Maximilian März, *Let's enhance: A deep learning approach to extreme deblurring of text images*, arXiv preprint arXiv:2211.10103 (2022).

[33] Dimitris Tsiktsiris, Antonios Lalas, Minas Dasygenis, et Konstantinos Votis, *Improving passenger detection with overhead fisheye imaging*, IEEE Access (2024).

[34] Ultralytics, `https://https://docs.ultralytics.com/models/yolov8/#usage-examples`, 2024.

[35] ———, `https://docs.ultralytics.com/models/yolov9/#performance-on-ms-coco-dataset`, 2024.

[36] ———, `https://docs.ultralytics.com/models/yolov10/#performance`, 2024.

[37] ———, `https://docs.ultralytics.com/models/yolo11/#performance-metrics`, 2024.

[38] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, et Guiguang Ding, *Yolov10: Real-time end-to-end object detection*, arXiv preprint arXiv:2405.14458 (2024).

[39] CY Wang, IH Yeh, et HYM Liao, *Yolov9: Learning what you want to learn using programmable gradient information. arxiv 2024*, arXiv preprint arXiv:2402.13616.

[40] Thaddäus Wiedemer, Stefan Wolf, Arne Schumann, Kaisheng Ma, et Jürgen Beyerer, *Few-shot supervised prototype alignment for pedestrian detection on fisheye images*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 4142–4153.

[41] Nicolai Wojke, Alex Bewley, et Dietrich Paulus, *Simple online and realtime tracking with a deep association metric*, 2017 IEEE international conference on image processing (ICIP), IEEE, 2017, pp. 3645–3649.

[42] Shidun Xie, Guanghong Deng, Baihao Lin, Wenlong Jing, Yong Li, et Xiaodan Zhao, *Real-time object detection from uav inspection videos by combining yolov5s and deepstream*, Sensors **24** (2024), no. 12, 3862.

[43] Lu Yang, Liulei Li, Xueshi Xin, Yifan Sun, Qing Song, et Wenguan Wang, *Large-scale person detection and localization using overhead fisheye cameras*, Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 19961–19971.

[44] M Yaseen, *What is yolov8: An in-depth exploration of the internal features of the next-generation object detector. arxiv 2024*, arXiv preprint arXiv:2408.15857.

Appendix A

Evaluation of the contribution of the internship and challenges

My internship at BusPas Inc. has been an enriching and transformative experience, significantly enhancing my technical skills and deepening my understanding of AI applications. Beyond theoretical knowledge, this opportunity allowed me to work hands-on in building and deploying an AI-driven pipeline for real-world scenarios, tackling challenges in computer vision, specifically in object detection using fisheye images, which is an area with limited existing literature.

The initial phase of my internship focused on conducting extensive research and literature reviews on computer vision techniques for object detection, with a particular emphasis on road object detection in fisheye images. This involved exploring the availability of open-source fisheye datasets containing key objects such as buses, cars, bikes, pedestrians, and trucks. Simultaneously, I had to familiarized myself with NVIDIA's DeepStream SDK, the TAO Toolkit, and the various modules available on the NVIDIA Jetson Orin Nano Development Kit, essential for real-time AI deployment.

As the project progressed, the focus shifted towards developing a robust object detection pipeline, fine-tuning models, and deploying them efficiently in the real-world setting of smart bus stops. One of the key aspects of this implementation was the extraction of crucial metadata, such as dwell time and object counting, for post-processing and data analysis. These insights would later contribute to improving urban mobility strategies and enhancing transit efficiency.

One significant challenge encountered during this internship was fine-tuning diffusion models to recreate overhead fisheye scenarios, particularly for detecting anomalies such as garbage and graffiti. The approach I followed was using LoRA and DreamBooth to fine-tune a Stable

Diffusion model tailored for anomaly detection. However, after several weeks without achieving promising results, the project was reassigned to another member of the Computer Vision team. At that point, I transitioned to working on the license plate detection pipeline, where I focused on building a robust dataset, optimizing object detection models, and integrating the solution into the overall AI pipeline.

Throughout the internship, collaboration with both the Computer Vision and Development teams was constant and effective. Regular discussions, feedback loops, and teamwork played a crucial role in refining the pipeline, troubleshooting deployment issues, and ensuring the seamless integration of the AI models into BusPas Inc.'s infrastructure. This experience not only strengthened my technical expertise but also enhanced my ability to work in a multidisciplinary team and deploy AI solutions in practical, real-world scenarios.

Appendix B

Timeline

Date	Description of Work
May to June	Conducted initial research and literature review on computer vision techniques for object detection, specifically for road object detection task using fisheye images. Also explored the existence of open source fisheye datasets that contain the desired objects: Bus, Car, Bike, Pedestrians, Trucks. At the same time, started to get familiarized with Deepstream, NVIDIA TAO toolkit, and the modules available at the Nvidia Jetson Orin Nano Development kit.
July to August	Implemented a dataset preprocessing pipeline to clean and structure raw data for model training. Developed a C++ script within DeepStream to extract metadata from the object detection model, specifically to calculate dwell time, and store the dataset in an SQLite database. Additionally, annotated Test Set 1 and designed a data augmentation pipeline for pedestrians, leveraging pedestrian masks by using the SAM model from open-source datasets to integrate them into real site images. Furthermore, started the training process of YOLO models with the first batch of data set, which was composed mainly with open source images.

Date	Description of Work
September to October	Conducted a literature review on fine-tuning diffusion models using Low-Rank Adaptation (LoRA) on diffusion models to generate synthetic fisheye images resembling real-site bus stops. Initially experimented with the Hugging Face Transformers package, but the results were unsatisfactory. As an alternative, explored COMFYUI for inpainting with Stable Diffusion, specifically to generate graffiti on advertisement panels of the bus shelters. Additionally, began a literature review on license plate detection and OCR-based character extraction. Also labeled the second batch of images collected from newly installed devices.
November to January	Trained YOLO models (versions 8 to 11) to build a benchmark using different combinations of available datasets and evaluated their performance on two test sets collected from distinct real-site locations. Additionally, developed a pipeline to remove pedestrians from real-site test images due to privacy concerns. This was essential to retain the dataset for long-term use while preserving the background and other objects such as cars and buses. Furthermore, utilized the OWLv2 model to label license plate images from real-site data to train a license plate detection model. After evaluating multiple open-source OCR models—both general character recognition models and those specifically designed for license plates—built a pipeline for car and license plate detection, followed by OCR processing for character extraction.

Appendix C

Dataset descriptions

The table below provides a detailed description of the datasets utilized in the experiments discussed in **Chapter 5**:

Name	**Description**
Dataset 1	Combination of the first batch of data, as described in **Section 3.1**, that consists of open source datasets and some real site images.
Dataset 2	Combination of the first batch of data with the images from the methodology followed in **Section 4.4** and also with the data augmentation on cars and buses.
Dataset 3	Combination of the new images collected from different real sites with the real ones collected in the first batch, including augmentation techniques as described in **Section 4.3**. This dataset also includes the open-source Fisheye8K images and CEPDOF. (5 classes).
Dataset 4	Combination of new images collected from real sites with the first batch without data augmentations. Only contains 4 classes since "Truck" was merged into the "Car" class.
Dataset 5	Combination of new images collected from real sites with the first batch, including data augmentations. Only contains 4 classes since "Truck" was merged into the "Car" class.
Dataset 6	Combination of real-site images with the first batch and open-source dataset CEPDOF for pedestrians with data augmentations. Contains 4 classes since "Truck" was merged into "Car".

Name	**Description**
Dataset 7	Combination of real-site images with the first batch, including data augmentations. Also includes images from the methodology followed in **Section 4.4**. Contains 4 classes since "Truck" was merged into "Car".
Dataset 8	Combination of real-site images with the first batch, including data augmentations. Also includes "Bikes" from the WoodScape dataset to handle class imbalance. Contains 4 classes since "Truck" was merged into "Car".
Dataset 9	Combination of real-site images with the first batch, including data augmentations, where the pedestrians are pixelated. Contains 4 classes since "Truck" was merged into "Car".

Table 11. Description of the combination of datasets used to train object detection models.