A Privacy-Preserving Edge Computing Solution for Real-Time Passenger Counting at Bus Stops using Overhead Fisheye Camera

Pardis Ghaziamin

A Thesis

in

The Department

of

Concordia Institute of Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

December 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Pardis Ghaziamin**

Entitled: **A Privacy-Preserving Edge Computing Solution for Real-Time Passenger Counting at Bus Stops using Overhead Fisheye Camera**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Farnoosh Naderkhani

_____ Examiner
Dr. Manar Amayri

_____ Supervisor
Dr. Nizar Bouguila

_____ Co-supervisor
Dr. Zachary Patterson

Approved by _____
Chair
Department of Concordia Institute of Information Systems Engineering

_____ 2023

_____
Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

A Privacy-Preserving Edge Computing Solution for Real-Time Passenger Counting at Bus Stops using Overhead Fisheye Camera

Pardis Ghaziamin

Successful transit planning in smart cities requires automated and efficient passenger counts at bus stops while also respecting the privacy of passengers—a paramount consideration in the age of responsible AI. In this thesis, we describe the implementation and development of a real-time passenger counting system at bus stops on Nvidia Edge devices powered only by solar panels, with limited memory, while not compromising privacy or incurring substantial costs. Numerous studies have developed and applied computer vision people detection techniques, although this has not been applied and optimized explicitly to edge-computing passenger counting at bus stops. In this regard, we evaluated different object detection models using a novel dataset from an overhead fisheye lens camera of passengers at a bus stop that we developed to analyze and improve the accuracy of the passenger counting system. We also optimize and reduce the models to allow them to be deployed on edge devices. We find that YOLO-V4 with mAP of 87% outperforms DetectNet-V2 and Faster-RCNN. The best object detection model then has been optimized and deployed on the Nvidia Jetson Device and the performance and efficiency of the passenger counting system evaluated. Deployment via Nvidia DeepStream on the edge showcased a more than 50% reduction in GPU, CPU, and memory consumption, enhancing efficiency while conserving energy. As a result, we present a more accurate and efficient edge-computing video analytics solution for an ethically responsible passenger counting system at the smart city bus stop.

Acknowledgments

I would like to express my heartfelt gratitude to my supervisor, Dr. Nizar Bouguila, whose guidance, support, and expertise have been invaluable throughout this research journey. Their insightful feedback, encouragement, and unwavering dedication have profoundly shaped this thesis. I am also deeply thankful to my co-supervisor, Dr. Zachary Patterson, for their mentorship and encouragement. Their expertise in the field, along with their continuous support and encouragement, has played a pivotal role in the successful completion of this work. I am truly fortunate to have had the opportunity to learn from their wisdom and experience. Their guidance has been instrumental in refining my research and improving the quality of this thesis. I am indebted to both of them for their patience, encouragement, and belief in my abilities. Their contributions have been immeasurable, and I am profoundly grateful for their support. I would like to convey our heartfelt appreciation to BusPas Inc., and Mitacs Accelerate Program for the support they have dedicated to this research. Additionally, I want to express my gratitude to Dr. Wissem Maazoun, VP of Innovation at BusPas where I had the privilege of interning. His support and encouragement during my internship period were instrumental in helping me gain practical insights into my field of study. The experiences I gained under his supervision were invaluable, and I am truly appreciative of the opportunities provided to me during my time at BusPas. I also extend my sincere gratitude to my parents, whose nurturing, trust, and unwavering moral and financial support have been instrumental in fostering my growth and enabling me to flourish.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Based on the World Urbanization Report by the United Nations Department of Economic and Social Affairs in 2018 [3], the population portion who are located in urban areas will increase to 60% by 2030 and 68% by 2050, while it was only 54% in 2014, despite the short respite in this trend due to COVID-19 [4][5]. Urbanization will amplify the challenge of traffic and transportation congestion in the cities. Moreover, based on the press release from the World Health Organization [6], nine out of ten people in the world breathe polluted air. This study found that exposure to tiny airborne particles causes about 7 million deaths worldwide annually. According to a report issued by Équiterre in 2009 [7] on the effects of transportation on the environment and public health, 40% of Quebec's greenhouse gas emissions are related to transport, with 82.7% of those emissions from the road transport industry. Solutions to reduce air pollution, noise pollution, landscape transformation, and climate change include the use of integrated mobility, electrification of transportation, and public transportation.

Public transportation, particularly bus travel, has the potential to decrease the use of private cars, reduce fuel consumption, and alleviate traffic congestion. Indeed, it is not only waiting times and travel duration but also the level of crowding inside buses that holds paramount importance for the efficiency of Public Transport. Overcrowding can dissuade individuals from opting for public transportation, thereby rendering accurate passenger counting a pivotal challenge in the context of smart cities.

Moreover, active modes, carpooling, and other innovative options might be able to address these

challenges effectively. Currently, transit agencies are able to count people onboard buses and other transportation means. The central issue lies in our limited understanding of passenger numbers at bus stops. Furthermore, accurately understanding the behavior of individuals who arrive at these stops but alter their plans due to prolonged waiting times for alternative transportation methods remains a significant challenge. This uncertainty poses a considerable hurdle in comprehensively assessing and managing passenger behaviour at these crucial transit points. Understanding passenger behaviour at bus stops can help stakeholders better plan, and passengers better choose, more sustainable ways to travel. Having an accurate number of passengers who are waiting at the bus stops can help transportation managers and urban planners better manage travel behavior, reduce passenger congestion, and improve public transportation services [8].

The research described in this work is related to work done with the smart bus stop company, BusPas [9], in order to establish an end-to-end people counting system (with overhead fisheye cameras) to their Nvidia GPU-powered "SCiNe" smart bus stop sign to improve the public transportation system. BusPas is a tech company that is developing a smart display named SCiNe which stands for "**S**mart **Ci**ty **Ne**twork", and is based on Nvidia Jetson devices with various IoT sensors, such as motion detection, temperature, display temperature, safety light, speakers, a fisheye camera and an e-paper screen. E-paper is a display technology that replicates the appearance of ink on paper, providing high contrast, wide viewing angles, and low power consumption by reflecting ambient light, making it ideal for devices with limited battery. To increase mobility and track efficiency, this device gathers telemetry data using a variety of sensors. All of these devices are connected to the cloud-based platform called ORA, which was created by the BusPas team, to simplify the communication and management of these devices. The SCiNe has a display to bring real-time information to passengers and is equipped with a camera positioned at 8 feet to have an optimal view of the bus stop. They aim to have the most optimal and efficient algorithm to be able to process on the edge device powered by solar panels. Figure 1.1 demonstrates a graphical schema of the close-up of the bottom of the SCiNe where the camera is located.

Recent reports have shown that the popularity of video-based automatic passenger counting systems (APCS) is on the rise [10][11]. Nowadays, data received from camera-based automatic passenger counting (APC) systems is a source of knowledge for solving a wide variety of urban

transport issues, ranging from public service improvements to real-time responses to critical traffic situations. Understanding these types of data while preserving privacy is fundamental to the development of smart cities [12]. Especially when it comes to edge computing where respect to the privacy and sustainability of the system are vital factors. Optimal edge computing systems play a vital role in developing smart cities due to the cost, energy efficiency, and privacy. Different from conventional AI methods, Edge AI technology merges Artificial Intelligence and Edge Computing. In this approach, AI algorithms and models are integrated into edge devices, enabling them to function locally without relying on internet connectivity. This eliminates network latency issues and privacy risks in signal transmission.

Furthermore, data security and privacy are important in passenger counting for several reasons:

- **Protecting passenger privacy [13]**: Passenger counting systems may collect data from passenger devices, such as laptops, smartphones, and tablets, to estimate the number of passengers.
- **Preventing data breaches [14]**: Passenger counting systems may collect sensitive data, such as passenger information, which can be used to identify potential security threats.
- **Ensuring data accuracy [15]**: Passenger counting systems rely on accurate data to provide useful insights into passenger behavior and ridership information.
- **Maintaining system integrity [16]**: Passenger counting systems are critical components of public transportation systems, and any disruption or compromise to the system can have serious consequences.

Therefore, it is important to ensure that the system is secure and that appropriate measures are taken to prevent system failures or attacks. Edge computing can enhance security and privacy by reducing the need to send data to the cloud since it processes data closer to the source without recording and storing it [12].

Nowadays, computer vision is being used worldwide, from security solutions to passenger counting in public transport [17]. Computer vision needs a capable camera to be used for passenger counting in public transportation. However, passenger counting is a complex task due to

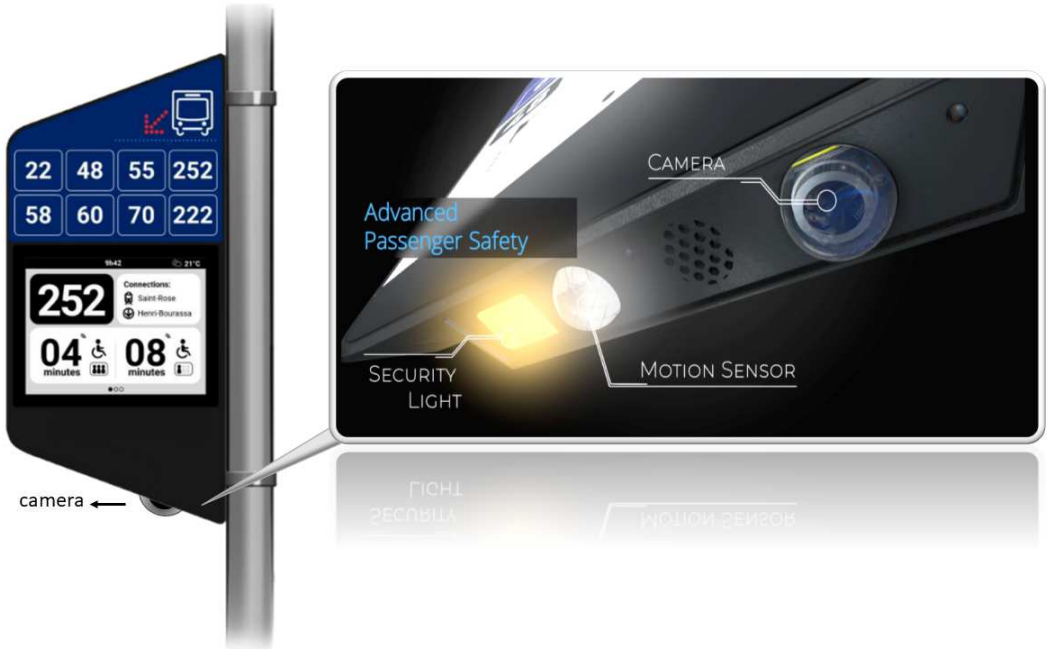

Figure 1.1: BusPas SCiNe Device (Left) and the Camera position underneath of the SCiNe (Right)

the variability of bus passengers in terms of their physical features and attire. Each bus stop has a different background, and lighting conditions can affect signal quality. Moreover, the duration of passengers getting on a bus is also very brief, lasting from 1 to 5 seconds, with an average of 2 seconds. Additionally, having high frames per second (fps) processing speed is essential because it provides more frequent data points, capturing rapid changes accurately and enhancing the system's responsiveness [18]. To address these challenges, using a capable camera with a wide-angle lens or mounting the camera higher can be a feasible solution [19]. Over the past two decades, numerous studies have been conducted on crowd monitoring. However, conventional cameras are limited in their ability because they cover a small field of view and have large blind areas. This results in the need for several cameras to be installed if one wishes to capture a 360-degree field of view. Additional cameras, of course, lead to additional (and higher) costs.

In contrast, fisheye lenses offer an unparalleled field of view, eliminating blind spots and providing a comprehensive perspective of the surroundings, making them more suitable and economical for people detection than standard lens cameras. Furthermore, overhead images offer a solution to the occlusion problem encountered with standard cameras [20], enhancing privacy by eliminating the need to capture facial information, and, at the same height top-view (which is 8 feet for the

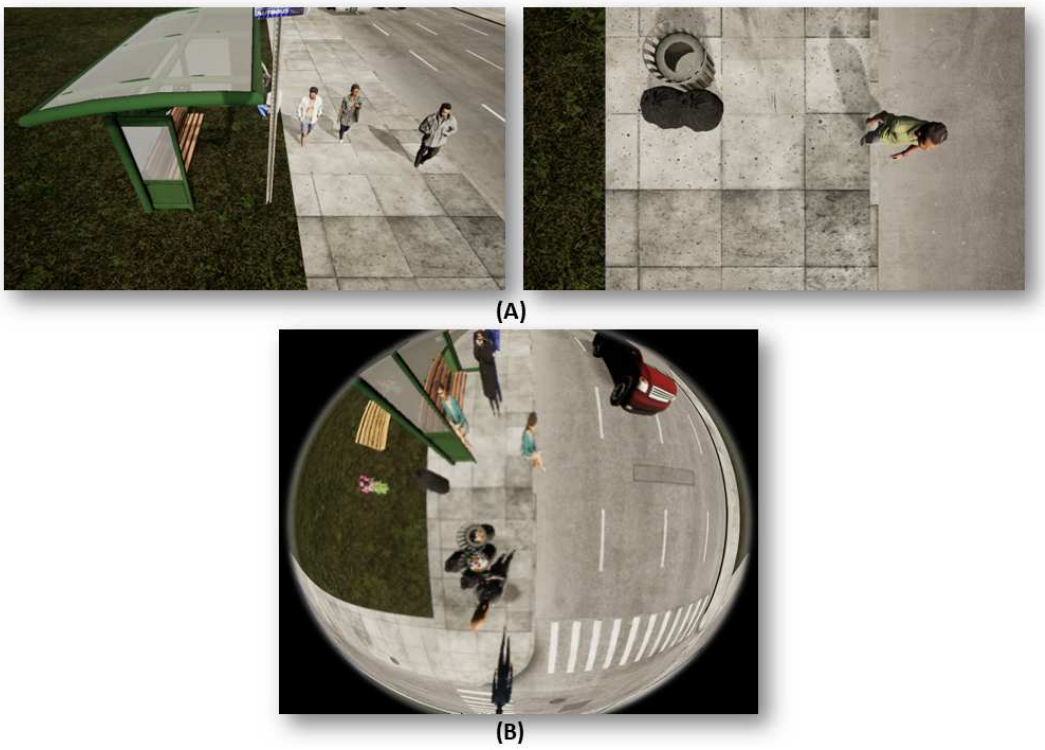

Figure 1.2: Overhead fisheye camera images vs standard camera images

BusPas SCiNe), fisheye cameras bring a greater field of view compared with standard cameras.

Figure 1.2 shows the field of view of each camera that was simulated by the AIGO simulation platform, presented by Rexys company [2]. Figure 1.2 (A) is the area that is captured by a standard camera in 8 feet height from two different perspectives, overhead and front, and Figure 1.2 (B) shows the area that fisheye camera can capture when is installed overhead in the same height. The comparison clearly demonstrates that fisheye camera brings a greater field of view at the same height which makes it a better fit for the passenger counting at the bus stop.

This 360-degree vision, combined with optimized edge computing, propels passenger counting into a realm of unprecedented accuracy and real-time analysis. Moreover, fisheye cameras bring about greater advantages when it comes to edge processing due to generally being smaller and more cost-effective [21]. While fisheye cameras have been used in intelligent robot and car safety device applications, little research has used them for people counting in crowd monitoring and no research has been conducted on passenger counting using fisheye cameras on the edge in public transportation areas [22]. Furthermore, there is no publicly available dataset of fisheye cameras of

passenger counting in an uncontrolled environment such as a bus stop.

Over the past two decades, numerous studies have developed and applied computer vision people detection techniques, although these studies have not been optimized explicitly for passenger counting at the bus stop, especially on the edge without relying on cloud-based solutions. Moreover, current models were developed using images from conventional, standard-lens cameras. While fisheye cameras have been used in intelligent robot and car safety device applications, only little research has used them for people counting in crowd monitoring and no research has been conducted on passenger counting using fisheye cameras on the edge in public transportation areas [22]. Furthermore, there is no privacy-friendly dataset of fisheye cameras of passenger counting in an uncontrolled environment such as a bus stop.

Traditional passenger counting systems have often been challenged with efficiency, especially in dynamic and densely populated environments. Fisheye cameras, with their panoramic view and unmatched depth perception, have revolutionized the landscape of surveillance and data collection. When coupled with the real-time processing prowess of edge computing, these fisheye cameras become not just lenses capturing moments, but gateways to a new era of intelligent transportation.

Besides Edge-AI, Deep Learning based object detection is also the main focus of this study. Deep Learning allows a machine to learn to detect or classify objects using a training dataset. Therefore, the performance of the model will depend on many factors of the trained data source, such as the quantity of images, the accuracy of the labels, or the angle at which the camera is positioned when collecting data. At the same time, Nvidia provides a platform, DeepStream, that supports the most common and powerful object (and therefore people) detection models. DeepStream is an end-to-end platform with pre-developed models for video analysis that can be trained with the Nvidia TAO (Train, Adopt, and Optimize) Toolkit. This is helpful because the Nvidia models are robust, having been trained on massive datasets. Moreover, Nvidia makes it possible to adapt these models with user-provided data, giving users the ability to take advantage of these well-trained models for their own tailored purposes. Finally, it allows for the optimization of the models to ensure they are as computationally efficient as possible because of the close integration of the edge accelerator, especially when it comes to real-time edge processing in Nvidia Jetson devices where there is limited memory and battery power due to the use of a solar panels. It turns out, however, that, as with

object detection more broadly, none of the Nvidia-implemented models have been used with fisheye cameras either. In this research, we evaluate different object detection models implemented with Nvidia DeepStream for passenger counting at bus stops with overhead fisheye cameras and deploy the best model into the Nvidia Jetson device to make it more privacy-preserving to process on the edge implement the passenger counting system on BusPas smart bus stop SCiNe.

The speed of processing in parallel for multi-sensor people detection systems has always been a challenge. Utilizing GPU time and memory efficiently requires significant effort and parameter tuning for object detection inference. This study focuses on scaling up the people detection and counting System using the Nvidia DeepStream SDK to make it more robust in processing multiple camera feeds simultaneously and generating real-time insights [23]. It gives us the benefit of deploying our models offline on the edge, which in turn is helpful as it provides real-time processing and also protects the privacy of the public at the bus stops, as we are not storing any images or videos. Instead, the camera feed is input to the device, which does the processing in real-time with the GPU on our device, and gives the results without having to store any images of people and invade their privacy. Deep stream is an end-to-end platform for video analysis that brings about many advantages, as it is able to work in an integrated manner with both input and output. It performs more effectively and efficiently than a deep learning model since it can process data independently and asynchronously better in various applications such as semantic segmentation, object detection, classification, and detection [24]. The Nvidia Deep Stream platform also provides the Nvidia TAO Toolkit (Train, Adopt, and Optimize) to train and adopt pre-trained models together with any custom dataset to perform well in any particular domain. The structure of the TAO Toolkit is shown in Figure 1.3 [25].

The Nvidia Transfer Learning Toolkit has been applied for model adoption and launch to the edge because of the potential benefits that can be gained from the tight integration of the edge accelerator and optimization [26]. The Nvidia Transfer Learning Toolkit (Nvidia TAO), primarily known as Nvidia TLT (transfer Learning toolkit) is a framework for training, adapting, and optimizing computer vision and conversational AI models with any custom data, in a fraction of the time without large training data sets or AI expertise. The TAO application layer is designed and built based on CUDA-X, which includes all the lower-level Nvidia libraries. These libraries include the Nvidia

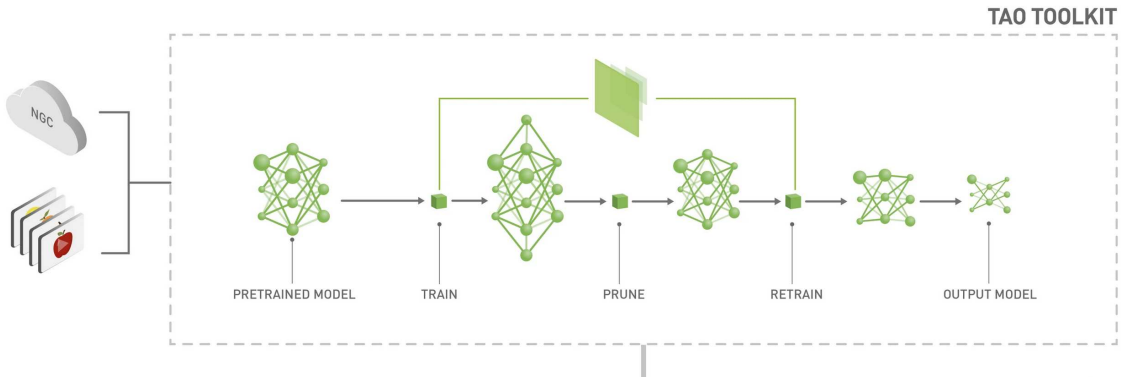

Figure 1.3: TAO Toolkit Schema [1]

Container Runtime for GPU acceleration, CUDA and cuDNN for deep learning (DL) operations, and TensorRT (the Nvidia inference optimization and runtime engine) for model optimization. As a result, the TAO Toolkit models are fully compatible with, and accelerated for, TensorRT, ensuring the best inference performance with no additional work. It actually allows the customization of Computer Vision (CV) and AI models by integrating Nvidia pre-trained models with your own data using the Nvidia TAO Toolkit [1]. One of the applicable pre-trained models for people detection is PeopleNet. PeopleNet is a specialized model designed for detecting pedestrians and other objects in the context of autonomous driving. It uses a combination of computer vision and deep learning techniques to accurately detect objects in real-time. According to Nvidia, PeopleNet can achieve inference speeds of up to 200 frames per second (FPS) on Nvidia's Jetson platform [27]. Besides purpose-built models, Nvidia TAO also provides training on some famous detection architectures, such as YOLOv3, FasterRCNN, SSD/DSSD, and RetinaNet some of which we will experiment the performance of. In this study, we have utilized state-of-the-art deep learning models such as PeopleNet, Detectnet-V2, YOLO-v4, ssd, Retina net, faster-RCNN.

Our contributions in this research involve three main parts which are first, collecting, processing, and annotating a rare privacy-friendly overhead fisheye dataset of passengers in a bus stop in an uncontrolled environment, second, training, adopting, and evaluating different Nvidia object detection models on the aforementioned dataset, and third, optimizing, and deploying the trained model into the Nvidia edge devices to assess the efficiency of implementing the passenger counting system.

The study achieved an end-to-end solution that integrates multiple live feeds through a single

application, making the people counting pipeline lightweight and improving processing speed on a low-end device like Jetson. The DeepStream SDK also allows for easy configuration and adaptation to new GPU environments with minimal effort. This study details the process used to achieve the solution and explores improvements in deploying the people-counting system on Jetson devices, which is discussed in further sections.

The next section provides background literature relevant to the research. Next comes a description of the methodology and the dataset. A subsequent section describes the experiments conducted followed by the study results and a concluding section.

Chapter 2

Literature Review

In the world of responsible AI and computer vision, accurate passenger counting stands as a fundamental necessity. Beyond its fundamental role in optimizing public transportation and enhancing customer experiences in retail, precise passenger counting has a vital role in informed decision-making and sustainable practices. By harnessing innovative technologies, such as overhead fisheye cameras and advanced AI algorithms, passenger counting not only ensures efficient resource allocation but also fosters safety, security, and privacy. As we delve into the depths of existing literature, it becomes evident that responsible implementation of these technologies is paramount, not just for operational efficiency, but also for upholding ethical standards and environmental conservation. This literature review aims to explore the intersection of responsible AI, computer vision, and passenger counting, shedding light on the transformative potential of these advancements in shaping a smarter, safer, and more sustainable future.

In this literature review, we conduct an extensive review of existing literature, exploring responsible AI solutions for passenger counting and the latest advancements in computer vision models for people detection, which form the foundation of our counting method.

2.1 Responsible Edge-Computing Object Detection

Several computer vision models have been developed for large GPU processing. However, sending data to the cloud is neither sustainable nor cost-effective for real-time streaming processing [28].

Moreover, it poses privacy risks during data transmission. In order to handle privacy issues, studies such as [29][30] focus on data anonymizing like blurriness or distortion that can impact accuracy and efficiency. However, encryption methods, while highly secure, pose inherent privacy risks because every encryption algorithm must have a corresponding decryption method. This duality is essential for legitimate users to access encrypted information. However, it also means that if unauthorized parties gain access to the decryption key or method, they can potentially compromise the encrypted data, thus challenging the confidentiality and privacy of the information. Therefore, even though encryption is a powerful tool for securing data, the potential for decryption underscores the need for additional layers of security and careful key management to mitigate privacy risks effectively [31]. Furthermore, encryption, while effective, can sometimes be computationally intensive, impacting system efficiency. Balancing security and efficiency becomes paramount, requiring careful consideration of encryption algorithms and key management practices to minimize privacy risks without compromising performance [32]. To address these issues, we propose the following: performing all processing on the edge, eliminating privacy risks associated with cloud-based GPU data transmission [33] while using a sustainable alternative. The following paragraph explores existing literature on edge computing with Nvidia-based edge devices.

There are some studies that have adopted Nvidia deep learning models using the Nvidia transfer learning toolkit (TAO) and then optimized them to deploy on edge devices, particularly Nvidia Jetson devices. For example, Ho Chuen Kam and Tomio G. both have adopted DetectNet-V2 and Resnet50 models for face mask detection [25][26]. Umair et al. trained and investigated the performance of different Nvidia pre-trained models for Automated Plastic-Bag Contamination Detection [34]. Simon et al. used Mask R-CNN and analyzed performance by tuning hyperparameters for real-time segmentation of Neonates [35]. Abu Anas et al. trained and optimized DetectNet-V2 for vehicle license plate recognition [36]. These studies used Nvidia models and deployed on edge devices for real-time applications but none of them was used in the context of passenger counting and utilizing overhead fisheye camera video. In the following section, we delve into the cutting-edge advancements in computer vision models tailored specifically for people detection.

2.2 Computer vision techniques for Object Detection

In the realm of computer vision, object detection stands as a pivotal domain, crucial for applications ranging from autonomous vehicles to surveillance systems [17]. With the advent of deep learning, remarkable progress has been made in developing sophisticated models capable of accurate object detection [37]. However, the traditional focus has largely been on object detection in standard images captured by conventional cameras. In recent years, the emergence of fisheye images as a unique category has presented both challenges and opportunities in the field of computer vision. In this regard, this part covers the latest advancements in computer vision models for people detection which consists of two main parts: studies focused on images captured by standard cameras, and studies utilizing images captured by fisheye cameras. Through this approach, first, we can comprehensively analyze the advancements, challenges, and methodologies in object detection models designed for standard images, and second, critically evaluate the emerging trends and research gaps in object detection models specifically designed for fisheye images. By comparing these two categories, this review aims to provide insights into the evolution of object detection techniques in the context of varying image types and pave the way for passenger counting using overhead fisheye images.

2.2.1 Counting people in Standard Images

In this study, standard images refer to images taken by a standard lens camera, which produces a perspective (or "gnomonic") projection of reality, and most of the images captured in real life are of this type. Based on previous works, there are two ways of counting people in these kinds of images that will be elaborated in the next session [38].

Direct and Indirect methods

All kinds of counting algorithms try to take the vector of features from images, and then apply different computations on top of that. Basically, a vector of features could be a better input than the original values of pixels. The two types of people counting methods adopt different statistical

methods. The indirect method, which is based on a regression model that utilizes feature vectors to count the number of people and direct methods, that apply classification models on feature vectors to detect people in the images.

According to many studies [39], and [40], direct detection methods are more accurate in sparse crowds, however, indirect methods work better in dense crowds. Crowd refers to situations where people overlap with each other.

In the case of Montreal bus stops, crowds tend to be sparse, so we can continue with direct counting methods that have better performance in the sparse crowd.

As all direct models initially detect people, it is simple to calculate the count from the number of detections. In this instance, it is clear that the precision of the chosen person detector directly affects the precision of the count. In this regard, we aim to improve and advance a detection model that suits our specific application.

In this literature review, we will go through the state of the art of people detection models. The public datasets for people (and other object) detection used to train these models are initially presented in the section that follows. Since statistical models often benefit from training on related datasets and auxiliary tasks, we present object detection datasets in addition to the person detection datasets [40][41].

Datasets

Datasets for conventional images: We can list 4 datasets specifically for people detection in addition to the 4 primary datasets for object detection (Pascal VOC, ILSVRC, MS-COCO, and Open Images) (Caltech Pedestrian, CityPersons, EuroCity, CrowdHuman). A few of these datasets are used to train most object or person detectors. These 8 datasets can be outlined as follows:

- **Pascal visual object classes (VOC)** [42]: There are 27,000 objects in the 11,000 photos, which are divided into 20 classes and include people, various animals, various cars, and some furniture or accessories. An example of the dataset is demonstrated in figure 2.2 (A).

- **ImageNet** large scale visual recognition challenge (ILSVRC) [43]: The 200 classes consist of more than 500,000 images and more than 500,000 annotations. The 200 classes of ILSVRC

also contain classes similar to the 20 classes in Pascal VOC. An example of the dataset is demonstrated in figure 2.2 (B).

- **Microsoft Common Objects in Context (MS-COCO)** [44]: More than 200,000 photos with more than 500,000 annotations are included which are grouped into 80 classes. This dataset is more difficult than the previous ones since it has more overlapping, tiny objects. Many published models are trained and tested on this dataset. An example of the dataset is demonstrated in figure 2.2 (C).

- **Open Pictures** [45]: This dataset has 600 classes and almost 2 million photos, with 16 million annotations because of which it is by far the biggest online object detection dataset. An example of the dataset is demonstrated in figure 2.2 (D).

- **Caltech Pedestrian** [46]: It includes 350,000 annotations from 2,300 different people on 200,000 photos taken from videos. An example of the dataset is demonstrated in figure 2.3 (A).

- **CityPersons** [47]: Despite the fact that this dataset is smaller (5,000 photos with 35,000 annotations of people) than Caltech Pedestrian, it has 20,000 different individuals than the latter, which is almost 10 times more. An example of the dataset is demonstrated in figure 2.3 (B).

- **EuroCity** [48]: Over 50,000 photos with 238,200 annotations of persons are included in it. The photos were taken in roughly 30 different cities during the day and night in all four seasons. Since it encompasses a wide range of backgrounds and light intensities, it is a challenging dataset. An example of the dataset is demonstrated in figure 2.3 (C).

- **CrowdHuman** [49]: This dataset has over 15,000 photos with 339,600 different annotations of humans. This data set frequently contains individuals who partially or entirely overlap, with about 22 individuals in each image. An example of the dataset is demonstrated in figure 2.3 (D).

- **Kitti Dataset** [50]: Karlsruhe Institute of Technology provides an autonomous vehicle dataset that applies to different applications such as object detection, tracking, segmentation, etc. The

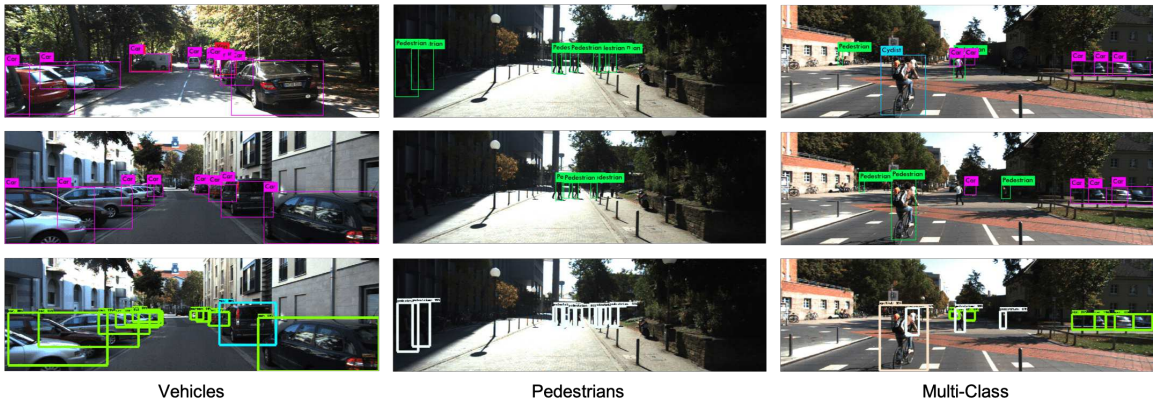

Figure 2.1: An example of public Object Detection Datasets with the standard camera: Kitti dataset

benchmark for object detection and object orientation estimation consists of 80.256 labeled objects spread across 7481 training images and 7518 test images. This dataset is captured by driving around the mid-size city of Karlsruhe. Per image, up to 15 vehicles and 30 people are observable. An example of the dataset is demonstrated in figure 2.1.

One feature links all of the aforementioned datasets: practically all of them consist of side-viewed photographs. Hence, models trained on these generalize on the distinctive visual characteristics of objects and individuals when viewed from the side. These models, however, cannot be relied upon to accurately identify humans in 360-degree diving-view images since they are not exposed to diving-view images. In the following sections, we describe a method to solve this issue for 360-degree photos taken with hypergone lens cameras that have vertical dive views. For the time being, we investigate object and person detectors in common photos. These detectors can be categorized as common or deep based on how they extract feature vectors from the images.

Classic detectors

If a detector extracts feature vectors from the images via manual calculations, it is referred to as a “classical” detector. The features that were extracted may indicate:

Key points: The combination of these spots that the individual considers special forms the feature vectors. They can be defined in a number of ways, including using Harris wedges [51],

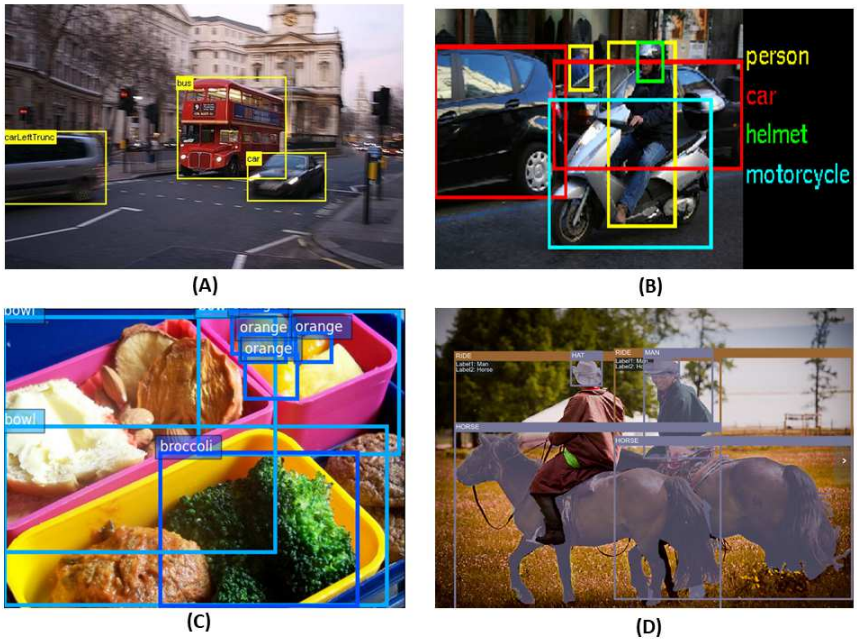

Figure 2.2: An example of public Object Detection Datasets with the standard camera: (A) Pascal visual object classes (VOC), (B) ImageNet, (C) Microsoft Common Objects in Context (MS-COCO), and (D) Open Pictures

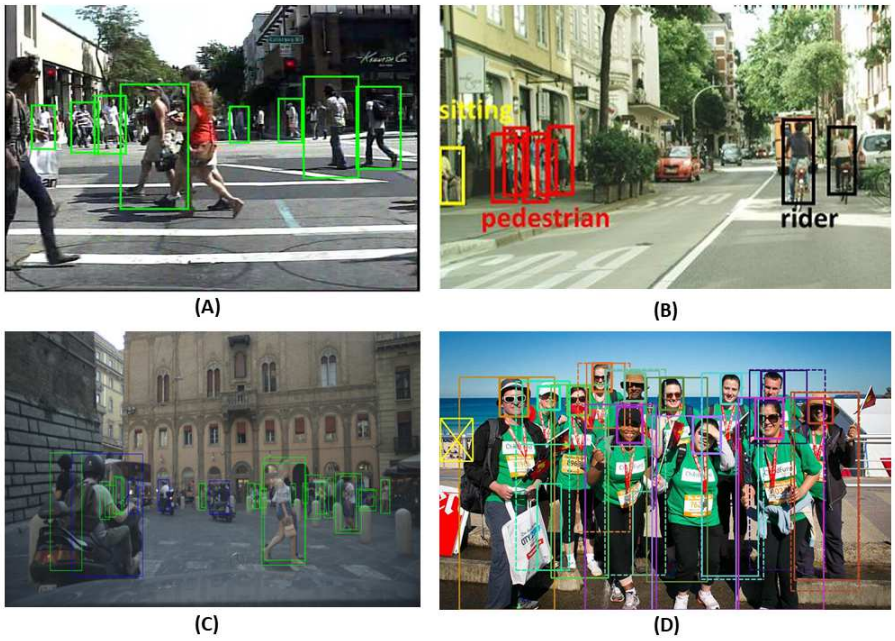

Figure 2.3: An example of public People Detection Datasets with the standard camera: (A) Caltech Pedestrian, (B) CityPersons, (C) EuroCity, and (D) CrowdHuman

SURF (speeded-up robust features) [52], or SIFT (scale-invariant feature descriptors) [53]. These descriptors are simple and quick to compute, but they also give persons encountered during the training phase a distinct identity. As a result, they are excellent at identifying people in new photos, but they have trouble identifying persons whose looks have not previously been learned [54]. So, key points are characteristics more suited to the task of re-identifying individuals rather than their detection and are not applicable to our study.

Texture indicators: There are a number of ways to extract texture indicators locally from images: using the local binary pattern (LBP) [55], with the histogram of oriented gradients (HOG) [56], with Haar wavelets [57], or with other methods. The histogram of image gradients is created by computing all of the gradients present in the image (using Sobel filters, for example), sorting the gradients on each small 8x8 neighborhood of the image into a histogram based on their direction and intensity, and then analyzing the histogram. These histograms for all neighborhoods are simply concatenated to create the HOG feature vector. HOG has been demonstrated to be a useful descriptor for person detection [56]. Yet, as will be discussed in the following section, the deep features learned from the data outperform HOG and other conventional descriptors at the detection task.

In several traditional detectors [56][57][58][59][60], the detection is carried out using the sliding window method. To establish whether a person is present at a certain area, they move a window to various scales and locations in the image, compute the chosen feature vector on this window, and then use a classifier (support vector machine (SVM) [61], AdaBoost [62], or another). There are modifications to this idea that increase precision or accelerate calculations. For instance, to better detect persons, the integral channel features (ICF) detector [58] converts images into several color spaces (HSV, LUV, grayscale) before computing the feature vectors. By providing a technique to approximate gradients on oversampled or undersampled images from the original gradients and avoiding repeating computations of feature vectors at various scales, the aggregated channel features (ACF) detector [59] speeds up computations. As a result, it can work two times faster than ICF without sacrificing precision.

The dynamic parts model (DPM) detector [60], which is a predecessor of deep convolutional networks for the detection of persons and objects, is worth mentioning at this point. A primary filter that encodes the object's overall appearance is used to model the objects that need to be detected,

together with higher resolution special filters that encode the object's numerous components and their ideal positions in relation to the object's center. These filters are used to weigh the features that were extracted from the images in order to detect objects, and a score is then given to each position in the images. It measures how well the primary filter matches that region of the images as well as how effectively the individual filters create high responses close to their ideal placements (a high value indicates the presence of an object at that location). Compared to ICF and ACF, the DPM detector generates more precise detections but is slower than ACF.

DPM is comparable to more current detectors based on deep convolutional layer networks since it uses filter-based object representations. DPM can also be expressed as a convolutional neural network(CNN) [63]. This research demonstrates an improvement in DPM performance by replacing deep features acquired from the data for a DPM's traditional features. This confirms the claim that manually derived features are weaker than deep features for detection. Because of this, we won't use a conventional detector to count people at bus stops; instead, the next part will offer models for item and person detection based on deep neural networks.

Deep Sensors

The extraction of feature vectors from images for a deep detector is now a process that a neural network has learned from the data rather than being manually engineered. Convolutional neural networks (CNN) are deep networks that perform best at detecting objects and people, which is potentially a result of the translational equivariance property of convolutions. During feature extraction, this feature enables the model to make use of identical parameters across the entire image. Several Convolutional model structures have been proposed, and we offer the top-performing models in this section in order of their date of development. As it is challenging to determine beforehand which of these models will be the most effective for our goal, we will employ a number of the deep detectors discussed in this part in our experiments to help us reach this determination. Moreover, deep sensors can be divided into two types: 1-step detectors, which find and classify objects in a single operation, and 2-step detectors, which propose regions of interest (RoI) on an image before classifying this RoI among the item classes taken into account.

2-step detectors

Initially, fast region-based convolutional neural networks (Fast R-CNN)[64] provide a classification layer for classifying all detections at once by integrating the regions of interest using the RoI pooling layer (taking as input an image and a list of regions of interest). This layer converts all areas of interest into feature maps of the same size, which makes them all processed at once by the classification network. The detector's previously extracted features separate for every region of interest, which significantly lengthens the computation time. Nevertheless, this article does not address the expensive stage in this detector known as the proposal of regions of interest.

Second, faster region-based convolutional neural networks (Faster R-CNN) [65] accelerate the proposal of regions of interest (RPN) with the aid of developing the region proposal network (RPN). A convolutional network called the RPN introduces the idea of anchor boxes. Taking into account the k anchor boxes, the RPN is made to predict k confidence scores and 4k relative dimensions at each image position. Following that, the RPN-proposed regions of interest are those with confidence levels above a certain threshold, and their dimensions are calculated in relation to the anchor boxes with the 4k predicted values. Combining RPN for proposing regions of interest with Fast R-CNN for classifying these regions of interest, helps the Faster R-CNN model to remain efficient and effective. Besides, RPN and Fast R-CNN have some shared convolutional layers that improve inference speed so that the final model is able to process 5-20 frames per second.

Lastly, the feature pyramid network (FPN) [66] is made up of convolutional layers that lower the feature maps' resolution and then convolutional layers that raise it (Figure 2.3) in which, these two feature maps are connected to each other with residual connection in the same direction. Moreover, links that are shortened (or residual) between feature maps in the same dimension are used. The resolution is then increased by making predictions from all the layers. Its architectural design makes it easier to predict regions of interest at various scales. Replacing the RPN for the region of interest proposal with an FPN in Faster R-CNN, the accuracy of the model can be increased by up to 10% and it can be slightly sped up.

1-step detectors

You Only Look Once (YOLO) [67] efficiently speeds up object detection by discarding the 2-step detection approach used by earlier detectors. The RPN deployed by Faster R-CNN has a similar structure to the YOLO model, while it does not make use of marker boxes. Instead, it uses a smaller grid size and does not make predictions anywhere on the image. Moreover, it predicts a confidence score and absolute box dimensions at each grid point. Because it operates directly on a picture and recognizes and locates objects inside it, it can be considered a comprehensive detection model. Although it achieves slightly worse performances than earlier versions, it operates at a rate at least twice as fast and respects the real-time limitation. The versions YOLO-V2 [68], YOLO-V3 [69], and YOLO-V4 [70] are then created by applying a number of advancements connected to the development of the deep learning sector to YOLO. The upgraded versions, for instance, predict multi-resolution detections as FPNs, raise the resolution of the input images, add normalization, use landmark boxes as RPNs, include residual connections, and lightly adjust the training (loss functions, activation functions, etc.). All of these changes enhance the model's accuracy without significantly slowing it down.

Similar to this, the single shot detector (SSD) [71] performs its predictions using bounding boxes and produces predictions on feature maps with different resolutions as opposed to a single grid which enables it to process 15 more frames per second while being 10% more accurate than YOLO. Nevertheless, since newer versions of YOLO use comparable designs with further improvements, SSD no longer stands out against these new versions.

Then, the RetinaNet model's developers [72] came up with the idea that the mismatch of training classes is what keeps one-step detectors from being as accurate as two-step detectors. They discover that "easy" instances including background areas that are exposed to one-step detectors much more frequently than "hard" ones including objects. Hence, they switch out the conventional cross-entropy classification loss function for the loss function. In summary, they introduce a parameter to cross-entropy costs that significantly reduces when the class probability is accurately predicted. Hence, the loss is more heavily influenced by "difficult" situations than by obvious ones. RetinaNet is now approximately 10% more accurate than SSD on the MS-COCO data set as a result.

2.2.2 Counting People in Fisheye Images

Relevant literature on people detection with fisheye camera images involves two aspects. First, computer vision models developed in this context, but also importantly, publicly available datasets that make model development possible. We start with publicly available fisheye lens labeled datasets.

Datasets for 360 degree images:

Since we are developing a model to work with overhead fisheye cameras in uncontrolled environments (i.e. bus stops), we require a suitable dataset. However, labeled fisheye lens camera datasets are extremely rare. Those that do exist include the following. The HABBOF dataset [73] comprises four brief videos with three or four persons in each. The MW-R dataset [74] is made up of frames that were taken from 19 films shot in indoor settings with a limited (1-5) number of (frequently the same) individuals between videos. The CEPDOF dataset [74] consists of 8 films with 8 to 13 persons in them captured in a break room of an office. The WEPDTOF dataset [75] is derived from YouTube videos. Overall, 188 different people can be found across all interior videos, with 1–35 people per film. The Woodscape dataset [76] is a fisheye dataset for autonomous vehicles captured by a car equipped with cameras. What is important to recognize here is that publicly available fisheye datasets are constrained primarily to the controlled overhead environments with few people, or uncontrolled environments but with point-of-view fisheye lenses mounted on vehicles. That is, there are no datasets from overhead fisheye cameras in uncontrolled environments, and obviously, none at bus stops either.

Labeled vertical dive view fisheye lens camera datasets are extremely rare. Few are annotated with encompassing right rectangles (around people) aligned with the axes of the images or with dots on the heads of people, such as MirrorWorld (collected and annotated by the Institute of Creativity, Arts, and Technology at Virginia Tech University), BOMNI [77], and PIROPO [78]. We do not use these three datasets in our research since their annotation formats are unfit for either training or evaluating detectors in 360-degree images. These datasets can be outlined as follows:

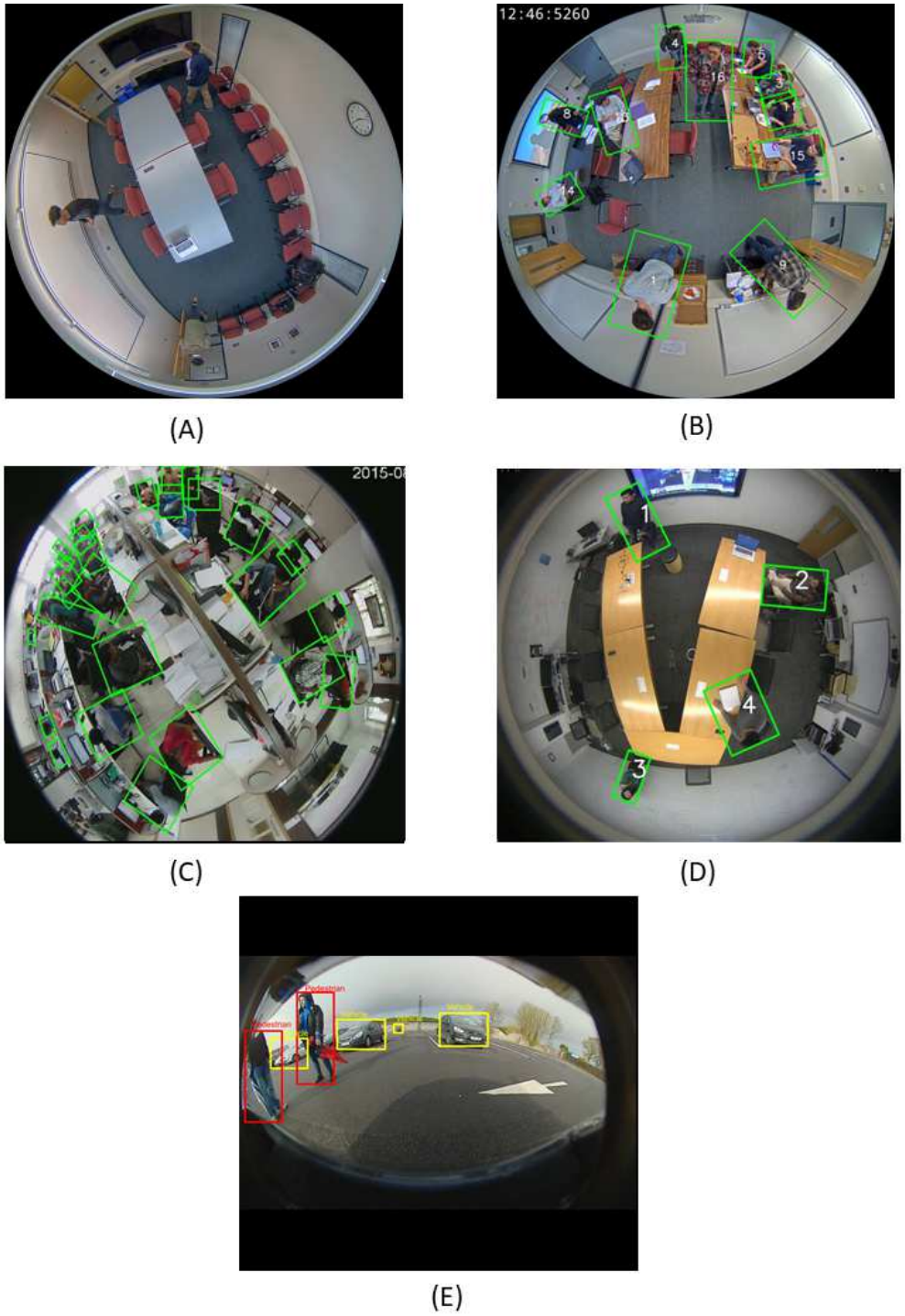

Figure 2.4: An example of public fisheye Dataset: (A) HABBOF, (B) CEPDOF, (C) WEPDOF, (D) MW-R, and (E) Woodscape

- **HABBOF: Human-Aligned Bounding Boxes from Overhead Fisheye cameras dataset** dataset [73]: It comprises of four brief videos with three or four persons in each. Due to its limited size, this dataset is not used to verify our models. An example of the dataset is demonstrated in figure 2.4 (A).

- **MW-R: Mirror Worlds Dataset** [74]: They updated the annotations on the Mirror-World dataset in 2020 and gave it the new moniker MW-R [54]. It is made up of frames that were taken from 19 films that were shot in indoor settings with a limited number of individuals present (1-5 people, frequently the same people from video to video). An example of the dataset is demonstrated in figure 2.4 (D).

- **CEPDOF: Challenging Events for Person Detection from Overhead Fisheye images** dataset [74]: It consists of 8 films with 8 to 13 persons in them, involving extreme overlaps between people, unusual human stances, poor lighting conditions, etc. and is designed to be a dataset for ”difficult” or ”challenging” situations. This dataset is utilized in the last part of our experiments to analyze the performance of the Detectnet-V2 model. An example of the dataset is demonstrated in figure 2.4 (B).

- **WEPDTOF: In-the-Wild Events for People Detection and Tracking from Overhead Fisheye Cameras** dataset [75]: This is made form YouTube videos. 188 different people can be found across all videos, with 1–35 people per film. Although there are still images of interiors, the settings are much more diversified (exhibition, warehouse, grocery store, daycare, office, etc.). An example of the dataset is demonstrated in figure 2.4 (C).

- **Woodscape** [76] [79]: Valeo Woodscape is a fisheye dataset for autonomous vehicles captured by equipped cars with cameras. It includes labels for various autonomous vehicle tasks such as semantic segmentation, monocular depth estimation, object detection (2D & 3D bounding boxes), visual odometry, visual SLAM, motion segmentation, soiling detection, and end-to-end driving (driving controls). It provides 1000 images with 2D bounding box labels including pedestrians, vehicles, bicycles, traffic lights, and traffic signs. An example of dataset is demonstrated in figure 2.4 (E).

Adaptation of structure of deep detectors

In this section, we discuss detection models that optimize their internal structure to take into consideration the equivariance of rotations in 360-degree vertical dive view photos. The AP_{50}, or alternatively the mAP is frequently used to assess the accuracy of detectors if only one class of objects (in our case, persons) is analyzed. To define this metric, we define its precision and recall as the ratio of correct predictions to the total number of detections and the ratio of true detections to the total number of people actually present in the image, respectively. Detection is often considered to be true if its intersection-over union with an annotation exceeds 50%. This lower bound is especially significant as the detections made by the predictor and both the precision and recall are dependent on it. The detector's precision-recall curve can then be plotted by tweaking it. The area under this curve is represented by the AP_{50}, and the mAP is the average of the AP_{50} across all classes taken into account. When these values are high, a detector is thought to be more effective.

Using a unique angle-aware loss function, the rotation-aware people detection (RAPiD) model modifies YOLO to predict encompassing rectangles centered around persons in images. An augmented MS-COCO is used to train this model, by adding rotations, reflections, scaling, and color changes to photos. It is then retrained using a cross-validation approach on the 360° MW-R, HABBOF, and CEPDOF image datasets (when retrained on MW-R and HABBOF, it is validated on CEPDOF, and vice versa). This model can process 7 frames per second on a GTX1650 graphics card and is quite accurate (97% AP_{50} on MW-R, 98% AP_{50} on HABBOF, and 86% AP_{50} on CEPDOF), but it needs to be trained on MS-COCO with the given data increments and retrained on 360 frames.

The model below, which Li et al. [80] described, is a multi-class detector for 360-degree pictures. They employ deformable convolutions to extract object properties from the warped 360-degree images, and their model's final predictions are irregular quadrilaterals, which appear to be the main two aspects that set it apart. To assess the effectiveness of their method, they compute the intersection-over-union of two asymmetric quadrilaterals using the Monte Carlo method. They simulate a 360-degree picture dataset by applying distortions to the Pascal VOC dataset in order to train and test their algorithm. For this dataset, which they dub VOC-fisheye, they find a mAP of

75%. Such a multi-class detector is not appropriate for our purposes, which is to explicitly count persons.

The authors of a different paper [81] contend that the incompatibility between trained fiducial boxes and the appearances of humans in 360-degree photos is what causes issues when utilizing a detector that has been previously trained on conventional images. So, they suggest an architecture (influenced by CenterNet) that, without the use of landmark boxes, determines the center of the detections and directly computes the size and orientation of the enclosing rectangles. Their model achieves an AP_{50} of 96% on MW-R, 96% on HABBOF, and 80% on CEPDOF when running at 20 frames per second on a GTX1070Ti graphics card.

There are few object detection models optimized for fisheye photos to detect people. Some traditional people detection methodologies such as HOG and LBP have been adapted to this purpose by changing the fisheye geometry of the images [82] [83][58]. For instance, Chiang and Wang [83] obtained HOG features from the top center of each fisheye image after rotating each image in tiny angular steps. They then used an SVM classifier to detect people. In a different approach, Krams and Kiryati [84] trained an ACF (aggregate channel features) classifier on standard images and restored the ACF features taken from the fisheye image for person detection. Recently, algorithms based on Convolutional Neural Networks (CNNs) have also been applied to this use case. By training a CNN on a rotated version of the COCO dataset [44], Tamura et al. introduced a version of YOLO that is rotation-invariant [67]. Their approach assumes that the bounding boxes in a fisheye image are parallel to the image radius during the inference stage. Another YOLO-based approach [85] uses undistorted versions of overlapping windows that were taken out of a fisheye image and then applies YOLO to them. Each fisheye image is rotated by Li et al. [86] in 15 steps, and YOLO is solely applied to the upper center region of the image, where subjects typically stand erect. They then use post-processing to eliminate duplicate detections of the same person. Their system applies YOLO, 24 times to each image, which makes it computationally complex even though it is quite accurate.

The rotation-aware people detection (RAPiD) model modifies YOLO, using an angle-aware loss function to predict rectangles around persons. It is trained on 360° MW-R, HABBOF, and CEPDOF datasets, and achieved 97% AP_{50}, 98%, and 86%, respectively [74]. In another study, Li et al. [80]

describe a multi-class detector for 360-degree pictures, employing deformable convolutions to extract object properties and predict irregular quadrilaterals. Their method achieves 75% mAP on their VOC-fisheye dataset, but it is built with a dataset that is not suitable for explicit person counting. Another paper [81] proposed an architecture inspired by CenterNet to address the incompatibility between trained fiducial boxes and human appearances in 360-degree photos. It directly computes the size and orientation of enclosing rectangles without landmark boxes. The model achieves 96% AP_{50} on MW-R, 96% on HABBOF, and 80% on CEPDOF.

In most of the cases, the datasets were neither comprehensive nor challenging enough (i.e., single object per image with no background noise). Moreover, none of the aforementioned models utilize a dataset that has been created in an outdoor setting, for detecting and counting passengers waiting at the bus stop. These critical analyses clearly suggested the need to develop a practical solution with challenging real-world data for identifying and counting passengers at bus stops.

2.3 Literature Review Summary

To summarize, In the area of computer vision and passenger counting technology, previous research has predominantly concentrated on large models demanding high computational resources for execution. Furthermore, most of the computer vision models utilized conventional cameras, leaving fisheye images largely unexplored. A significant gap persists in developing an efficient people detection model utilizing fisheye cameras, particularly in the context of edge computing, where safeguarding passenger privacy and ensuring sustainability are critical concerns. This unexplored territory highlights the need for innovative solutions in this domain.

Moreover, there are a large number of models available that could potentially be used to undertake passenger counting at bus stops. At the same time, Nvidia has a toolkit that has implemented the most common and powerful models and allows them to be estimated based on user data. These models have, for the most part, not been used with fisheye cameras. This is where we highlight our contribution to this research. Addressing this gap, our research stands out by employing an overhead fisheye camera on edge devices, a pioneering approach that conserves energy, upholds privacy, and eliminates the need for transmitting sensitive data to the cloud. By addressing these

critical challenges, our research offers a novel perspective and practical solution, marking a significant advancement in the domain of passenger counting technology. This innovative solution not only conserves energy, making it environmentally friendly, but also prioritizes passenger privacy, a paramount concern in public transportation settings. By eliminating the need to transmit sensitive data to external servers or cloud platforms, our approach ensures that passenger information remains secure and confidential. This not only aligns with current privacy regulations but also builds trust among passengers, making them more receptive to such technological advancements.

In essence, our research does not merely offer a technical solution; it introduces a holistic approach that combines cutting-edge technology with ethical considerations. By embracing the overhead fisheye camera, we redefine the standards of privacy-friendly passenger counting technology. This groundbreaking methodology not only marks a significant advancement in the domain but also sets a new benchmark for responsible innovation in the field of computer vision and edge computing.

Chapter 3

Methodology

This research focuses on people detection and counting at bus stops using overhead fish-eye images on edge devices. One of the challenges is acquiring a dataset that meets our needs. The subsequent step involves building an optimal model based on this dataset, enabling efficient detection and counting people at bus stops, and deploying and implementing a people counting system on Nvidia edge devices.

3.1 Data

This research focuses on the task of people detection and counting in overhead images. In this work, two major datasets were used, one of which is CEPDOF which was mentioned in section 2.2.2. It is known to be a difficult and challenging dataset for overhead images for people detection. The other dataset used for this study comprises images captured using an Nvidia-compatible camera module - IMX477, deployed on one of the SCiNe devices in our testing environment. The dataset encompasses approximately 2212 images, providing a diverse range of scenarios and crowd densities. As shown in the following image, a few representative photos from our dataset showcase the variations in lighting conditions, perspectives, and crowd compositions. The availability of this dataset enabled us to investigate and develop robust algorithms for accurate people detection and counting in overhead imagery.

Capturing our own images and creating a dataset was necessary for several reasons. Firstly,

our camera captured overhead fisheye images, which provide a unique perspective and field of view compared to traditional camera setups. Existing datasets for overhead images often focus on controlled environments, such as the CEPDOF dataset, which features images taken in an indoor office break room setting. However, our testing environment is uncontrolled and situated in an open outdoor space, introducing various factors that affect people detection, such as varying light conditions, which resulted in different shadow placements during different times of the day. In addition, on some days we experienced rain, and on others, dust accumulated on the camera lens resulting in blurry images. By capturing our own images, we ensure that our dataset reflects the challenges and complexities present in real-world scenarios.

Moreover, our dataset includes images captured during low-light conditions, such as nighttime or cloudy days. This adds another layer of difficulty for people detection algorithms, as visibility is reduced and distinguishing individuals becomes more challenging. Additionally, our dataset contains instances of occluded objects, where people may be standing beside the bus stop or partially obstructed, not in clear view of the camera. These occlusions are common in crowded areas and present a significant challenge for accurate people detection.

By creating our own dataset, we addressed the limitations of existing datasets, catered to our specific testing environment, and included a wide range of real-world conditions. This allowed us to develop and evaluate more robust and reliable algorithms for people detection in overhead images.

3.1.1 Capturing images

To capture the images for our dataset, we employed the camera module IMX477, renowned for its image quality and compatibility with Nvidia toolkit. The camera module allowed us to capture 360-degree overhead fisheye images, providing a comprehensive view of the surrounding area.

The images were captured at an average interval of 10 seconds, ensuring a consistent and continuous stream of data. This interval was chosen to capture the dynamics of the environment, including the movement of people and changing lighting conditions. Over a period of 15 days, we conducted image-capturing sessions ranging from 3 to 10 hours each day, resulting in a substantial collection of 17,529 images.

By collecting images over an extended duration and covering multiple days, we aimed to capture a diverse range of scenarios and crowd dynamics. This comprehensive dataset allowed us to study and address challenges associated with varying light conditions, temporal changes, and crowd density. The abundance of images also facilitates robust algorithm development and evaluation, contributing to the reliability and accuracy of people detection and counting in overhead imagery.

3.1.2 Data cleaning

To ensure the quality and relevance of our dataset, a thorough data-cleaning process was conducted. We focused on creating a dataset specifically tailored to our testing environment, which primarily involved the area in and around the bus stop.

The initial data comprised approximately 17,529 images captured over a span of 15 days, as stated before. To filter the data, we implemented a careful selection process. Our objective was to retain only those images that contained people within our area of interest, namely the vicinity of the bus stop. This filtering step aimed to eliminate images that did not directly contribute to our research objective of people detection and counting in the designated testing environment.

Through meticulous examination, we reviewed each image and identified those that captured individuals in the vicinity of the bus stop. After this filtering process, we obtained a refined dataset comprising approximately 2,200 images, which sufficiently represented the dynamics and scenarios occurring in our testing environment. This cleaned and curated dataset provides a focused and targeted resource for training and evaluating algorithms for people detection and counting in the specific area of interest around the bus stop.

3.1.3 Data Annotation

For the neural networks to be supervised trained, images with bounding boxes must first be provided. To annotate this dataset, we leveraged the powerful annotation platform called Roboflow. The annotation process began with manual annotation of a subset of images, where we carefully labeled the presence and location of people in the overhead images. This initial annotated data was then utilized to train a model using Roboflow's training capabilities, which developed the ability to predict people's positions in the remaining unlabeled images. To expedite the annotation process

for the rest of the dataset, we employed Roboflow's label assist feature. This feature utilizes the trained model's predictions as suggestions, which were then reviewed and corrected by us, ensuring accurate and efficient annotations for a large volume of images in our dataset. The combined efforts of manual annotation and the utilization of Roboflow's label assist feature resulted in a comprehensive and accurately annotated dataset for our research on people detection and counting in overhead images.

They were initially converted to KITTI format [87], after which the pictures were cropped to a 1280 * 720 size. They were then introduced to the data layer for additional processing. After the data feeding, online image augmentation was carried out to prevent overfitting and provide the AI with more training data.

3.2 Nvidia Pre-trained models

To find the most optimized solution, we have adopted and evaluated several variations of cutting-edge computer vision object detection models (such as detecnet_v2 Faster R_CNN, YOLOv4) to determine which model performed the best. The following is a presentation of the theoretical background for each of the implemented computer vision models [34]. The following object detection models were implemented via the Nvidia Tao toolbox, which is based on TensorFlow.

3.2.1 Detectnet_V2

DetectNet is a novel object detection framework developed by Nvidia by predicting the objectness (coverage) within grid squares. With this knowledge, bounding box coordinates may be found as they correspond to the centers of the grid squares that are occupied. The DetectNet architecture is outlined below:

-Data Ingestion and Augmentation Layer

-Fully Conventional Network (FCN)

-Loss Function

-Bounding Box Clustering

After preparing and augmenting the dataset, the fully convolutional network received the training data and used it to extract features and predict the location of bounding boxes and object classes. Coverage maps, which have a grid of cells to assess whether an object is present in patches on them, were used to do this. In order to determine the total system loss, two distinct loss functions—the bounding box loss and the coverage loss—were combined.

Equation (1) displays the coverage loss based on the difference between the output coverage map produced by the fully convolutional neural network and the one produced from the training data (ground truth), which is measured as the least square error (L2), where $coverage^t$ is the actual value obtained from the ground truth, $coverage^p$ is the system's predicted coverage value, and N is the batch size. The bounding boxes correspond to the second loss.

$$Coverage\_loss = \frac{1}{2N}\sum_{i=1}^{N}|coverage_i^t - coverage_i^p|^2 \tag{1}$$

Equation (2) demonstrates that the mean absolute difference (L1) between the ground truth and the output-predicted bounding boxes was used to determine the bounding box losses. (x_{min}, y_{min}), and (x_{max}, y_{max}) indicates the two points of the bounding box, and (x^t, y^t) refers to the ground truth, while (x^p, y^p) refers to the predictions of the model.

$$Bbox\_loss = \frac{1}{2N}\sum_{i=1}^{N}|x_{min}^t - x_{min}^p| + |y_{min}^t - y_{min}^p| + |x_{max}^t - x_{max}^p| + |y_{max}^t - y_{max}^p| \tag{2}$$

The bounding boxes generated by the predictor were clustered and filtered by DetectNet in the final layer. These bounding boxes were then grouped together based on their similar locations and sizes [26].

3.2.2 Faster_RCNN

Ren et al. [88] proposed Faster R-CNN model addresses the issue of high processing cost when computing on the region proposals. This model is based on a novel Region Proposal Network (RPN) that was designed with the purpose of sharing features coming from the feature extraction with detection networks in order to significantly cut down on computational costs.

Further, the Fast R-CNN and RPN networks were merged using the shared CNN features and introduced the attention-based mechanism. In the RPN, anchors are used to address the multiple scales and aspect-ratio problems related to objects. As a result of this operation, an anchor is placed at the center of each spatial window. The proposals are then parametrized in relation to the anchors. This results in a unified single model with two modules: the RPN deep CNN model and the Fast R-CNN detector. Compared to other object detection models, the proposed RPN network generates multi-scale anchors as regression and adopts a pyramid type approach to make it efficient. Therefore, the loss function includes both the classification and regression tasks as expressed in Equation (3). It can be observed that both the regression loss and classification loss are optimized to train the model.

Additionally, the Fast R-CNN and RPN networks were combined with the CNN features that they had in common and the attention-based mechanism was introduced. RPN utilizes anchors to handle to issues with multiple scaling and aspect-ratio issues related to objects . This process results in an anchor being positioned in the center of each spatial window. In reference to the anchors, the proposals are subsequently parametrized. The RPN deep CNN model and the Fast R-CNN detector are the two modules that result in a single, unified model. The proposed RPN network creates multi-scale anchors as regression and uses a pyramid-style method to increase efficiency compared to previous object detection models. As a result, the loss function, as described in Equation (3), takes into account both the classification and regression tasks. It is clear that the model's training minimizes both the classification and regression losses.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{bbox}}\sum_i p_i^* L_{bbox}(t_i, t_i^*) \tag{3}$$

where i is the index for the anchor, p_i refers to the probability for the i_{th} anchor, p_i^* is the ground truth for the i_{th} anchor, t_i is the vector containing the predicted bbox coordinates, t_i^* is the vector for the ground truth bbox coordinates, cls and $bbox$ refer to regularization parameters, and l is the balancing parameter.

The shared convolutional layers in the backbone network initially extract the deep features associated with people from the images during training. This network is known as the feature extractor. The information retrieved by the RPN layer will be used as input to the pooling layer alongside an image with the fixed size. The final step involves connecting a high-dimensional feature vector to an output detection network with fully connected layers. In the output network, one fully connected layer is utilized to determine the classification score, while the other layer is used to determine the position to be detected by regression. During the training process, the neural network's parameters are modified based on the loss function (see Equation (3)). The SGD optimizer modifies the network's weights based on the output of the loss function in order to reduce the model's loss during back propagation. The steps involved in this process are as follows:

First, the network's weights (w) and bias (b) are initialized depending on the backbone network; Once the forward-propagation process has begun, the input image will be processed based on the type of the network layer.

The following expression is used to do forward computation for a fully connected layer:

$$\alpha^{m,l} = \sigma(z^{m,l}) = \sigma(W^l \alpha^{m,l-1} + b^l), \tag{4}$$

where m stands for the image sample, l for the network layer, σ signifies the activation function (in this case, ReLU), W for the network weights, and b for the network bias;

The following expression is used to do forward computation for a convolutional layer:

$$\alpha^{m,l} = \sigma(z^{m,l}) = \sigma(W^l \otimes \alpha^{m,l-1} + b^l), \tag{5}$$

where $\otimes$ stands for the convolution operation;

For the pooling layer, a reduced dimension operation is applied on the input ;

A Softmax function is utilized to predict the class probabilities for the output layer. Mathematically, the softmax operation is expressed as:

$$Softmax(z)_j = \frac{exp^{z_j}}{\sum_{k=1}^{K} e^{z_k}} for j = 1, 2, ..., K, \tag{6}$$

where K is the z-vector's dimension, on which Softmax is being applied;

Depending on the layer in the network, a back propagation operation applies using the loss function. Weights and bias values are changed for each layer in accordance with the gradient values throughout the backpropagation process, which uses the gradient descent approach to minimize loss. The gradient descent method depends strongly on the learning rate, which must be properly selected during training. It was decided to use a learning rate of 0.02 for the Faster R-CNN training [34].

3.2.3 You Only Look Once Version 4 (YOLO-V4)

In order to achieve accurate and fast performance for mobile platforms deployed in the field for real-time applications, Bochkovskiy et al. [89] suggested the YOLOv4 model. YOLOv4 is frequently described as an enhanced version of YOLOv3 with more accuracy and speed. The new model has a variety of universal features that can be applied to enhance performance, such as Cross-Mini-Batch Normalizations (CmBN), Cross-Stage Partial Connections (CSP), mish activation, and Self Adversarial Training (SAT).

An optimized backbone architecture, a neck architecture, and a detecting head architecture make up the overall structure of YOLOv4. YOLOv4 by default was created utilizing CSPDarkNet53 as the backbone, an additional SSP module, a PANet neck model, and a YOLOv3 head model. The input is split in two by the CSPDarkNet53 backbone network; one portion passes through the

DenseNet network while the other does not. The main reason the SPP and PAN are used is due to their improved receptive fields. A max pooling operation is applied at the SPP layer, in order to get around the fixed size input restriction, leading to fixed output representations. Within the network, PANet performs the pooling operation at multiple layer levels in order keep the spatial information.

In the end, YOLOv3 head architecture applies for the objects' detection and localization. YOLOv4 introduced SAT and mosaic data augmentation methodologies and uses genetic algorithms to optimize the model hyperparameters in terms of improving training performance. The mosaic data augmentation method combines four training samples, eliminates the requirement for several mini-batches, and generates better object features. On the other hand, the training image is altered when using the SAT augmentation, and the model is then trained on the altered image to find objects of interest. Figure 4 depicts the YOLOv4 model's architecture. While sustaining real-time performance, the new YOLOv4 model performed better than the YOLOv3 model [34].

Figure 3.1: The Internal hardware connections

3.3 Hardware Design

The proposed passenger counting system is a part of bus stop sign presented by BusPas. It mainly consists of two hardware components:

- A camera to capture the video
- An edge-computer to process the video through the computer vision models to detect contamination.

For the developed prototype, Arducam Camera and a Nvidia edge-computer were used. This experiment has been done first on the labratory on Nvidia Jetson Nano Development kit and then Nvidia Jetson Orin AGX and the last experiment was running on the SCiNe which is equipped with Nvidia Jetson Orin Nano. Figure 1 shows the laboratory hardware setup for the proposed passenger counting system. Brief details of each hardware component are provided as follows:

- Arducam High-Quality Camera: BusPas signs are equipped with 12.3MP Camera Module with 180-degree fisheye camera Lens which is installed at the bottom of the SCiNe about 8 feet high to make sure that it doesn't capture any information about the faces of the people.
- Nvidia edge device: The edge device is a system-on-module supporting popular AI frameworks capable of running multiple neural networks simultaneously. It offers real-time inference of the people detection model and does not store any image. For the developed prototype, Nvidia Jetson Orin Nano, Nvidia Jetson Orin AGX, and Nvidia Jetson Nano edge-computers were used.

Figure 3.1 shows a schema of BusPas SCiNe design for our laboratory test setup with Nvidia Jetson Nano and Arducam fisheye camera.

Chapter 4

Experiment

4.1 Experimental Data

As stated in the previous sections, we captured our own overhead fisheye images in our testing environment which was crucial due to unique perspectives and uncontrolled outdoor testing conditions. Our dataset addressed limitations of existing datasets, featuring varying light conditions, occluded objects, weather restrictions like rain, accumulation of dust on the camera lens, and real-world complexities. Using the IMX477 camera module, we captured 17,529 images over 15 days, ensuring comprehensive coverage. Thorough data cleaning focused on the area, in and around the bus stop, resulted in a refined dataset of approximately 2,200 images. These images were then annotated to create labeled data for training and annotations were converted to KITTI format. This curated dataset enabled the development of robust algorithms for people detection and counting in the specific testing environment, considering crowd dynamics and diverse scenarios.

4.2 Augmentation

To further enhance the diversity and robustness of our dataset, we applied various data augmentation techniques. These augmentations aimed to introduce variations and simulate real-world

scenarios, ensuring that our trained models could generalize well to different conditions. The following augmentations were utilized:

1. Horizontal Flip: We performed a horizontal flip on the images, effectively doubling the dataset by mirroring it along the vertical axis. This augmentation is particularly beneficial in our scenario, where people can be present on either sides of the bus stop. By mirroring the images, we expanded the dataset to encompass both orientations, enabling our models to learn and recognize people from different perspectives. The Nvidia TAO Toolkit documentation provides further information on this augmentation technique [90].

2. Rotation: To introduce variations in the orientation of the images, we applied rotation augmentation. This technique randomly rotates the images by a certain degree, simulating different camera angles and perspectives. By incorporating rotation, we aimed to improve the model's ability to detect and count people from various viewpoints, mirroring real-world scenarios where individuals may be positioned at different angles relative to the camera.

3. Saturation, Contrast, and Brightness: We also employed augmentations to adjust the saturation, contrast, and brightness of the images. These modifications mimic changes in lighting conditions, ensuring our models can effectively detect people under varying illumination levels. By incorporating these augmentations, we aimed to improve the model's robustness to different lighting scenarios, such as low-light conditions during nighttime or the presence of shadows caused by sunlight.

By leveraging these augmentation techniques, we expanded and diversified our dataset, allowing our models to learn from a broader range of scenarios and improve their performance in real-world situations.

4.3 Evaluation

An automatic people detection system has been created using a conventional two-stage data-driven research methodology (see Figure 6). The data preparation stage, which is the initial step, involved sorting, filtering, and processing the raw images that were taken from our test environment. Then, at this step, photos were labeled using the Roboflow annotation tool. To satisfy the criteria

of the training module, the labels were converted to KITTI format. The second stage is known as the model training phase, when object detection models were selected and the hyperparameters for training were determined in accordance with the literature (e.g., Detectnet-V2, Faster R-CNN, YOLOv4). It was decided to use Resnet-18 [91] as the feature extraction backbone. It was selected because it balances accuracy and inference time, which is appropriate for this use case. It was pretrained using a subset of the Google OpenImages dataset prior to training [92]. And then, the deployment and optimization will be done to implement on edge devices. Figure 4.1 shows the workflow that is followed in this project.

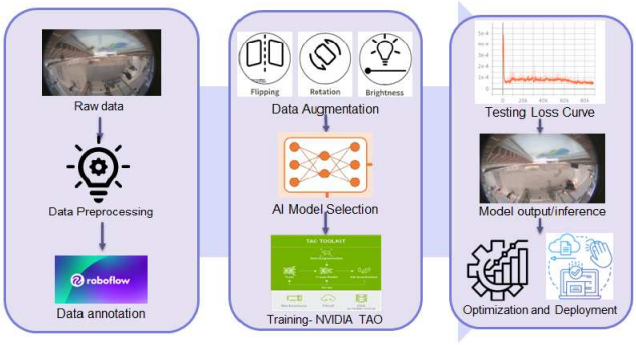

Figure 4.1: Block diagram representation of the research approach for people detection

The chosen models were adopted using the Nvidia TAO toolbox, and the performance was evaluated using the training loss, validation loss, and validation mAP values to make sure that the training followed the expected patterns. The trained models' predictions were tested and evaluated using numerous software and hardware performance metrics during the final stage, which is known as the testing and validation stage.

In this study, the Nvidia TAO toolbox (Train, Adopt, and Optimize), which is based on TensorFlow and Python, was used to train, adopt and then optimize the object detection model. All of the models were trained on the OMEN GT13-0090 30L Gaming PC with characteristics of Nvidia GeForce RTXTM 3090, Intel Core i9-10850K, and HyperX 32 GB DDR4- 3200 XMP MHz RAM (2 3 16 GB). For training, testing, and validation, a data split of 80%, 10% and 10% was utilized, respectively. Resnet18 has been used as the backbone of the networks. Each model was first trained with a batch size of 120, 80, 12 epochs respectively. Then, the models were pruned (with a pruning

threshold of 0.2 for Detectnet-v2, 0.2 for Faster R-CNN model and 0.1 for YOLOv4), and then re-trained for same numbers of epochs. The most optimal hyperparameters were determined through a process of experimentation. In order to lower the complexity or size of the model without compromising the integrity of the model as a whole, pruning is a popular strategy used in neural networks. This leads to improved memory consumption, reduced training time, and quicker inference times. However, the pruning threshold should be chosen carefully since it affects the accuracy. Because some significant weights may have been lost during the pruning process, a pruned model may see a decrease in prediction accuracy. In order to maintain accuracy after pruning, it is important to retrain the model. The Stochastic Gradient Descent (SGD) optimizer with 0.9 momentum and a base learning rate of 0.02 with L2 regularization with weight of 1e-4 was utilized for Faster R-CNN models. The learning rate with the minimum of 1e-7, and 4.99e-06 and the maximum of 1e-4, and 0.0005 was used for YOLO-V4 and Detectnet-v2, respectively. L1 regularization with the weight of 3.00e-09 and 3e-5 and the Adaptive momentum (Adam) optimizer was used for detectnet-v2 and YOLO-V4, respectively.

Using various matrices, the performance of the developed people detection system was evaluated. The testing and training phases each involved a separate evaluation of the software's performance. Using training loss, validation mAP, training time per epoch, and training curve monitoring, the training performance of computer vision models was assessed. The mAP was used to evaluate the test performance of models on the unseen validation dataset. The mathematical formula for calculating mAP is in Equation (9).

$$mAP = \frac{1}{N} \sum_{i}^{N} AP_i, \tag{7}$$

where AP stands for Average Precision, which is the weighted sum of precision at each threshold, with the weight corresponding to the increase in recall. One of the most widely used metrics for assessing the effectiveness of object detection models is AP, which is derived from the precision-recall curve. The standard in the Pascal VOC Challenge [42], which measures the AP from the outcomes with the Intersection Over Union (IoU) greater than 0.5, is followed by the average precision per class and the overall mean Average Precision. N stands for the number of classes. In this

instance, mAP is equivalent to AP because there is just one detection class (i.e., people). Precision can be calculated as the equation below, where TP and FP refer to the number of true positives and false positives respectively.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$\tag{9}$$

Both the validation and testing datasets were used to evaluate models. The evaluation of the developed people detection system is presented in this part. For each evaluation, the results are presented numerically, visually, and qualitatively to emphasize the key trends.

4.4 Experimental Results

The training and testing capabilities of object detection models for people detection were assessed. Tabular representations of quantitative results are provided, and training curves are used to display the enhancement.

Inference speed and model size are one of the most important factors to take into account because this people detection system is a real-time application on the edge where we have memory and power limitations. Some of the trained model's parameters have been pruned to reduce the size of the model while preserving its structure. The detailed impacts of pruning on computer vision detection models are quantitatively presented in Table 4.1. The table shows the size reduction of each model after pruning based on the pruning threshold. Because the pruning process might have lost some useful weight, it was retrained to regain its accuracy [26]. After pruning, the loss was seen to grow for a few epochs for all three models before decreasing to its minimal value. The loss of significant weights during the pruning phase caused the accuracy of the model to degrade, which was anticipated. However, after retraining, the pruned model was able to not only maintain but also have an increase in accuracy and less training loss.

Figure 4.2 shows the training loss curve for the DetectNet-V2. A negative exponential trend

was also observed for the training loss curve after pruning; besides earlier convergence, the loss kept on decreasing after the pruning of models which is an evidence of effective pruning. Validation loss also has been observed and showed a similar behavior however early stopping technique was considered to avoid overfitting.

Table 4.1: Models' size comparison before and after pruning

	DetectNet-V2	YOLO-V4	Faster-RCNN
before prune	45MB	243MB	102MB
after prune	22MB	23MB	51MB

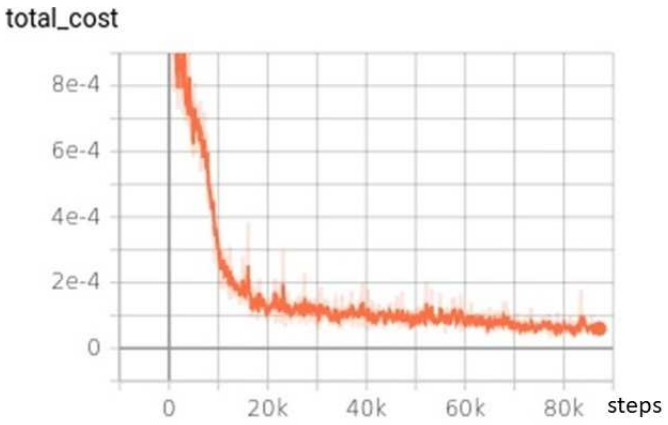

Figure 4.2: Training loss curves for Detectnet-V2 before and after pruning

Our algorithm evaluation comprises three sections. In each section, we employed the object detection models: first, with our custom dataset; second, with the augmented dataset that accounts for environmental challenges. In the final section, we examined DetectNet-V2 model's performance on the CEPDOF dataset (high-activity scenarios) and we compared these results with the RAPID model [74], which was introduced alongside the CEPDOF dataset.

.

The trained computer vision models were subjected to both validation and testing datasets to evaluate their test performance. The test performance of implemented models was compared based on the mAP values. For smooth and real-time monitoring, the optimal model should have high average precision to avoid false positives and false negatives as well as low inference time / high

frame rates. As it is shown in Table 4.2, training and adopting each of the models with our created dataset, YOLO-V4 with the mean average precision of 86.88% outperforms detectnet-v2 and faster-RCNN. The mAP train and retrain refers to the mean average precision of the validation set during the first training and retraining after pruning.

Table 4.2: Models' Performance comparison with our dataset

Model	(mAP)train	(mAP)retrain	(mAP)testing
DetectNet-V2	92.85	92.28	75.01
YOLO-V4	88.42	89.42	**86.88**
Faster-RCNN	96.95	96.60	86.60

In the second experiment, we used data augmentation techniques to expand and enhance our dataset so that the object detection model would be more robust to the possible environmental challenges that might occur such as rain, dust, and different lighting conditions. Table 4.3 demonstrates the performance comparison of each model on this augmented dataset. As it is shown, the testing mAP for each model has been increased and YOLO-V4 still is outperforming other models in precision.

Table 4.3: Models' Performance comparison with our Augmented dataset

Model	(mAP)train	(mAP)retrain	(mAP)testing
DetectNet-V2	80.06	79.69	77.05
YOLO-V4	88.76	89.28	**87.23**
Faster-RCNN	96.96	99.18	86.83

In the last experiment, we analyzed the performance of Detecnet-v2 on the CEPDOF dataset compared with the RAPID object detector model presented with CEPDOF. The architecture of the model is elaborated in the background part. CEPDOF dataset has different scenarios and in this experiment, we only considered the "High-activity" scenario. As Table 5 indicates, the testing performance of adopted detectnet-v2 after pruning with the mAP of 98.27% excels the RAPID model trained on CEPDOF, High-activity scenario, with the mAP of 94.2% . Given this experiment, the performance of the best-performing Nvidia model was observed to be comparable to the literature when a similar challenging real-world dataset has been used and signifies the effectiveness of Nvidia object detection models.

Table 4.4: DetetcNet-V2 Performance comparison with RAPID model

Model	(mAP)train	(mAP)retrain	(mAP)testing	(mAP)RAPID testing
DetectNet-V2	98.03	98.23	**98.27**	94.2

Nonetheless, it can be seen from the mAPs that, following model pruning, the model trained by the augmented dataset outperformed the others in terms of performance. It was observed that the greatest mAP of almost 87% was attained by the YOLO-V4 demonstrates a greater ability to be used as our people detector. Figure 4.3 shows the synthetic image of how our testing environment looks like. The model has been tested in a real environment that can not be shared due to confidentiality and privacy. Our tests on the real environment show the ability of the model to detect both further and closer people with different views and in different scenarios proving that this system also works with different orientations of humans to the camera, and it is robust to shadowing or different lighting and weather conditions. Moreover, our system is doing the whole process real-time without recording any image, and only the number of passengers will be sent to cloud in the every given time interval (for example 5 minutes).

Figure 4.3: Synthetic images of our test environment [2]

4.5 System setup

In this step, we explain how to implement the passenger counting system. First, trained people detection models have been optimized with TensorRT engine to reduce the model size and optimize hardware resources. This was achieved by consolidating and removing redundant layers in the models. Optimized people detector models then were combined with Nvidia object trackers in order to complete the counting task. Nvidia has several trackers such as NvDCF, NvSORT, and NvDeepSORT after experimenting with them we found NvDeepSORT performing better on this use case.

In order to distinguish between the passengers and pedestrians, a region of interest (ROI) has been defined which helps to count only people who are staying at the vicinity of the bus stop shelter. This way helps us to only consider passengers who are waiting for the bus and the number of passengers (RF) will be sent to the cloud for further analysis.

Furthermore, we used TensorRT engine as the optimizer to decrease the model size and resources of the hardware by merging and eliminating duplicated layers in the models. TensorRT engine optimizes deep learning models and allows them to run with high-performance inference. TensorRT is compatible with popular frameworks in deep learning today such as PyTorch, TensorFlow, Caffe, Darknet. By converting the darknet pretrained file of YOLOv4 to trt file of TensorRT, the speed of YOLOv4 is increased to 30 FPS from 9.5 FPS on Nvidia Jetson Orin AGX while the accuracies are similar to the unoptimized models. It optimizes the workspace memory by reducing memory footprint and reusing by allocating the memory when the tensor is used in its period and preventing the memory allocation overhead. As of the final experiment, to magnify the strength of the solution, we have experimented with other people detection implementations as well. We have deployed YOLO-V4 once via DeepStream and optimized with TensorRT and another time without DeepStream and with Darknet framework to evaluate the efficiency of DeepStream on Nvidia Jetson devices. Table 4.5 shows the strength of the implemented YOLO-V4 model via DeepStream over the performance of the same model, implemented via the darknet. YOLO-V4 implemented with DeepStream allocated 776MB of GPU, 42% of the CPU and 241MB memory compared with the YOLO-V4 implemented with darknet that allocated 1.6GB of GPU, 128% of CPU, and 657MB

of memory which is more than two times of consumption when we implement the object detection model via DeepStream. CPU usage of more than 100% refers to when it takes multiple CPUs. This shows the computational efficiency of the DeepStream models optimized with TensorRT which is vital for edge processing tasks, given the constraints in power and memory resources.

Moreover, YOLO-V4 is only able to process 9.6 FPS (frame per second) when running without DeepStream, but it is able to run 30 FPS when it is implemented through DeepStream, which is considered an important factor in our comparison because real-time processes require a high FPS for smooth performance.

Table 4.5: Computing comparison on Jetson Orin

	GPU	CPU%	Memory	FPS
YOLO-V4 (running via DeepStream)	776MB	42.0	241MB	30
YOLO-V4	1.6GB	128.8	657MB	9.5

Chapter 5

Conclusion

Effective transit planning and operations are aided by passenger counting, which is affected by numerous factors. The use of deep learning models for detecting and counting passenger flows is becoming increasingly common. However, having an optimal view of passengers at the bus stop and analyzing that while preserving privacy presents a challenge. Overhead fisheye cameras offer an advantageous perspective of the bus stop area. Yet, most of the people detection deep learning models have been developed on the images captured by standard images. Additionally, this process traditionally has been hindered by the reliance on heavy deep learning models requiring significant computational resources such as large GPUs or cloud-based GPUs, where there is a high risk of compromising privacy.

In this study, we presented a passenger counting system on edge devices, without any need to transmit data to the cloud, using computer vision at bus stops to optimize passenger flow and transit vehicle efficiency, reducing waiting times. We developed a privacy-friendly overhead fisheye dataset of passengers at bus stops and utilized Nvidia DeepStream and Transfer Learning Toolkit (TLT), leveraging pre-trained models retrained and pruned on our dataset. Three deep learning models, DetectNet-V2, YOLO-V4, and Faster-RCNN, were analyzed, comparing their accuracy (mAP). Our results show that YOLO-V4 adopted on our augmented dataset outperforms the other two models. We also deployed the adopted object detection model into the Nvidia Jetson Orin and evaluated the performance of real-time passenger counting models implemented via DeepStream.

The result shows the YOLO-V4 running via DeepStream has less than half GPU, CPU, and memory usage on the edge compared to the same model running without DeepStream.

The experimental findings have confirmed the good performance and practicality of the proposed object detection and tracking system for real-time monitoring scenes. The methods of this research are cost-effective and stable compared to the conventional approach for detecting objects in real time by hardware and do not involve the development or installation of any large-scale work on existing monitoring machines. Furthermore, the appearance of Jetson Orin is a new step for machine learning technology, since there won't be a server to receive the information from the edge to process the frame and make decisions. Instead of using a standard camera to get the information, the cameras are fisheye to be more cost-effective and smaller and to bring a greater field of view. These cameras are located overhead to capture bigger views while not capturing any facial information and respecting the privacy of passengers. This system has become more intelligent through the integration of deep learning models on Jetson devices. As Edge-AI, these devices can perform tasks efficiently and effectively, eliminating the need for a large server to process inputs or store output results. On the other hand, our dataset contains images captured by overhead fisheye cameras from passengers waiting at the bus stop in our test environment which is a rare dataset.

For future work, the model's precision, efficiency, and reliability can be improved by having access to additional data sets with a wider time range. Future work would also consider more patterns representing exogenous dependencies, like particular weather conditions and traffic incidents. Other aspects of extending this research would be considering the influence of network parameters on model performance. Moreover, even though the practicability of the model is verified through a bus case study, the applicability of the model for other scenarios like bike flows, and migration flows, could be examined.

In essence, our work not only presents a technical breakthrough but also embodies a transformative shift in how we approach real-time monitoring and data analysis. By blending cutting-edge technology with practicality, privacy consciousness, and cost-effectiveness, we have paved the way for a new era of responsible AI applications, with far-reaching implications for various industries and societal advancements.

Bibliography

[1] "NVIDIA TAO Toolkit." https://developer.nvidia.com/tao-toolkit. NVIDIA, Accessed: January 3, 2023.

[2] "Aigo." https://rexys.io/. by Rexys Inc., Accessed: October 15, 2023.

[3] "United nations, department of economic and social affairs, population division, world urbanization prospects 2018: Highlights." https://population.un.org/wup/, 2018. Accessed: October 10, 2022.

[4] S. A. M. Saleh, S. A. Suandi, and H. Ibrahim, "Recent survey on crowd density estimation and counting for visual surveillance," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 103–114, 2015.

[5] "COVID19." https://www.clevelandfed.org/en/newsroom-and-events/publications/cfed-district-data-briefs/. Accessed: October 2, 2023.

[6] "World health organization, 9 out of 10 people worldwide breathe polluted air, but more countries are taking action." https://www.who.int/news/, 2018. Accessed: October 23, 2023.

[7] "Transport, environment and public health, greenhouse gases (ghg) and climate change." https://www.equiterre.org/fr/ressources/fiche-transport-environnement-et-sante-publique, 2009. Accessed: October 23, 2023.

[8] A. Baghbani, N. Bouguila, and Z. Patterson, "Short-term passenger flow prediction using a bus network graph convolutional long short-term memory neural network model," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2677, p. 036119812211126, 08 2022.

[9] "Buspas inc." https://buspas.com/. Accessed: Novomber 20, 2022.

[10] C. Pronello and X. R. Garzón Ruiz, "Evaluating the performance of video-based automated passenger counting systems in real-world conditions: A comparative study," *Sensors*, vol. 23, no. 18, 2023.

[11] Dataintelo, "Global automatic passenger counting system market, by type (ir beam, thermal imaging, video based, others), by application (railway system, highway system, others), and by region (north america, latin america, europe, asia pacific, middle east amp; africa), forecast from 2023 to 2031." https://dataintelo.com/report/automatic-passenger-counting-system-market/. Accessed: Ocrober 20, 2023.

[12] G. Delzanno, L. Caputo, D. D'Agostino, D. Grosso, A. H. Mustajab, L. Bixio, and M. Rulli, "Automatic passenger counting on the edge via unsupervised clustering," *Sensors*, vol. 23, no. 11, 2023.

[13] "Edge security and privacy." https://journalofcloudcomputing.springeropen.com/edgesecurityprivacy. Accessed: August 23, 2023.

[14] "Anonymous automated passenger counting on public transport." https://parquery.com/98-accurate-anonymous-passenger-counting-on-swiss-trains/. Accessed: August 23, 2023.

[15] "People counting on the edge for accurate real-time results on axis cameras." https://facit.ai/insights/. Accessed: August 23, 2023.

[16] Y. Abdelwahed, O. Khaled, A. Elhamahmi, M. Dessouki, and K. Badawi, "Privacy-centric ai-based real-time storage-less edge computing approaches for passenger counting and action

classification on public transport vehicles," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 3116–3121, 2021.

[17] E. Dilek and M. Dener, "Computer vision applications in intelligent transportation systems: A survey," *Sensors*, vol. 23, no. 6, 2023.

[18] P. C. Madhusudana, X. Yu, N. Birkbeck, Y. Wang, B. Adsumilli, and A. C. Bovik, "Subjective and objective quality assessment of high frame rate videos," *IEEE Access*, vol. 9, pp. 108069–108082, 2021.

[19] P. Lengvenis, R. Simutis, V. Vaitkus, and R. Maskeliunas, "Application of computer vision systems for passenger counting in public transport," *Elektronika ir Elektrotechnika*, vol. 19, 09 2013.

[20] M. Ahmad, I. Ahmed, K. Ullah, I. khan, A. Khattak, and A. Adnan, "Person detection from overhead view: A survey," *International Journal of Advanced Computer Science and Applications*, 2019.

[21] "Fisheye security cameras: What they are, the pros,and the cons." www.tridon.com/fisheye-security-cameras-what-they-are-the-pros-and-the-cons. Accessed: October 23, 2023.

[22] X. Hu, H. Zheng, Y. Chen, and L. Chen, "Dense crowd counting based on perspective weight model using a fisheye camera," *Optik*, vol. 126, no. 1, pp. 123–130, 2015.

[23] S. Gupta, P. Gupta, and S. Kumar, "Robust multi-sensor facial recognition in real time using nvidia deepstream," *International Journal of Engineering and Technical Research*, vol. 11, 01 2022.

[24] N. Abdulghafoor and H. Abdullah, "Real-time moving objects detection and tracking using deep-stream technology," *Journal of Engineering Science and Technology*, vol. 16, pp. 194–208, 02 2021.

[25] T. Goto and M. Hongo, "Improvement of face recognition accuracy for mask wearers," in *2022 IEEE International Conference on Consumer Electronics - Taiwan*, pp. 301–302, 2022.

[26] X. Kong, K. Wang, S. Wang, X. Wang, X. Jiang, Y. Guo, G. Shen, C. Xin, and Q. Ni, "Real-time mask identification for covid-19: An edge computing-based deep learning framework," *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 01 2021.

[27] H. Feng, G. Mu, S. Zhong, P. Zhang, and T. Yuan, "Benchmark analysis of yolo performance on edge intelligence devices," *Cryptography*, vol. 6, no. 2, 2022.

[28] G. Premsankar, M. Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 02 2018.

[29] Z. Qi, A. MaungMaung, Y. Kinoshita, and H. Kiya, "Privacy-preserving image classification using vision transformer," *ArXiv*, 2022.

[30] P. He, C. Griffin, K. Kacprzyk, A. Joosen, M. Collyer, A. Shtedritski, and Y. M. Asano, "Privacy-preserving object detection," *ArXiv*, 2021.

[31] Stefanie.babb, "Top 5 encryption threats you need to know, comodo ssl resources." https://comodosslstore.com/resources/top-5-encryption-threats-you-need-to-know/, 2023. Accessed: November 2, 2023.

[32] X. Cheng and Z. Liang, "Data encryption standard based protection method of privacy data on internet of things," in *Proceedings of the 2016 4th International Conference on Machinery, Materials and Computing Technology*, pp. 1284–1290, Atlantis Press, 2016/03.

[33] G. sun, X. Xing, Z. Qian, and W. L. Li, "Edge computing assisted privacy-preserving data computation for iot devices," *Computer Communications*, vol. 166, pp. 208–215, 2021.

[34] U. Iqbal, J. Barthelemy, P. Perez, and T. Davies, "Edge-computing video analytics solution for automated plastic-bag contamination detection: A case from remondis," *Sensors*, vol. 22, no. 20, 2022.

[35] S. Lyra and S. Rira, "Real-time segmentation of neonates on embedded gpus via deep learning," in *POSTER Prag 2022*, 2022.

[36] A. A. I. Samad and T. A. Prema, "Recurrence sorting method for improved accuracy of unconstrained fast-moving vehicle license plate recognition system," in *2022 5th International Conference on Artificial Intelligence for Industries (AI4I)*, pp. 23–26, 2022.

[37] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, pp. 1–21, 01 2019.

[38] A. Dorais, "Dénombrement et localisation automatisés de passagers aux arrêts d'autobus avec une caméra," Master's thesis, Polytechnique Montréal, December 2022.

[39] G. Gao, J. Gao, Q. Liu, Q. Wang, and Y. Wang, "Cnn-based density estimation and crowd counting: A survey," *ArXiv*, vol. abs/2003.12783, 2020.

[40] Z. Fan, H. Zhang, Z. Zhang, G. Lu, Y. Zhang, and Y. Wang, "A survey of crowd counting and density estimation based on convolutional neural network," *Neurocomputing*, vol. 472, pp. 224–251, 2022.

[41] S. Ruder, "An overview of multi-task learning in deep neural networks," *CoRR*, vol. abs/1706.05098, 2017.

[42] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, 2014.

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, 2015.

[44] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV*, 2014.

[45] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The open images dataset v4," *International Journal of Computer Vision*, vol. 128, pp. 1956–1981, mar 2020.

[46] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 304–311, 2009.

[47] S. Zhang, R. Benenson, and B. Schiele, "Citypersons: A diverse dataset for pedestrian detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4457–4465, 2017.

[48] M. Braun, S. Krebs, F. Flohr, and D. M. Gavrila, "Eurocity persons: A novel benchmark for person detection in traffic scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1844–1861, 2019.

[49] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, and J. Sun, "Crowdhuman: A benchmark for detecting human in a crowd," *ArXiv*, vol. abs/1805.00123, 2018.

[50] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[51] C. G. Harris and M. J. Stephens, "A combined corner and edge detector," in *Alvey Vision Conference*, 1988.

[52] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), (Berlin, Heidelberg), pp. 404–417, Springer Berlin Heidelberg, 2006.

[53] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–, 11 2004.

[54] G.-A. Bilodeau, "course notes cse8770," 2020. G.-A. Bilodeau, Polytechnique Montréal, November 2020, course notes CSE8770.

[55] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[56] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, 2005.

[57] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I–I, 2001.

[58] P. Dollár, Z. Tu, P. Perona, and S. J. Belongie, "Integral channel features," in *British Machine Vision Conference*, 2009.

[59] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, 2014.

[60] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[61] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifier," *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, vol. 5, 08 1996.

[62] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *International Conference on Machine Learning*, 1996.

[63] R. Girshick, F. Iandola, T. Darrell, and J. Malik, "Deformable part models are convolutional neural networks," 2014.

[64] R. Girshick, "Fast R-CNN," *2015 International Conference on Computer Vision*, 2015.

[65] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[66] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017.

[67] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 779–788, IEEE Computer Society, jun 2016.

[68] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, 2016.

[69] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *ArXiv*, vol. abs/1804.02767, 2018.

[70] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *ArXiv*, 2020.

[71] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*, pp. 21–37, Springer International Publishing, 2016.

[72] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

[73] S. Li, M. O. Tezcan, P. Ishwar, and J. Konrad, "Supervised people counting using an overhead fisheye camera," in *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2019.

[74] Z. Duan, M. O. Tezcan, H. Nakamura, P. Ishwar, and J. Konrad, "Rapid: Rotation-aware people detection in overhead fisheye images," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.

[75] M. O. Tezcan, Z. Duan, M. Cokbas, P. Ishwar, and J. Konrad, "Wepdtof: A dataset and benchmark algorithms for in-the-wild people detection and tracking from overhead fisheye cameras," in *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[76] S. Yogamani, C. Hughes, J. Horgan, G. Sistu, S. Chennupati, M. Uricar, S. Milz, M. Simon, K. Amende, C. Witt, H. Rashed, S. Nayak, S. Mansoor, P. Varley, X. Perrotton, D. Odea, and P. Perez, "Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, (Los Alamitos, CA, USA), pp. 9307–9317, IEEE Computer Society, nov 2019.

[77] B. E. Demiröz, Ari, O. Eroğlu, A. A. Salah, and L. Akarun, "Feature-based tracking on a multi-omnidirectional camera dataset," *2012 5th International Symposium on Communications, Control and Signal Processing*, pp. 1–5, 2012.

[78] C. R. del Blanco, P. Carballeira, F. Jaureguizar, and N. García, "Robust people indoor localization with omnidirectional cameras using a grid of spatial-aware classifiers," *Signal Processing: Image Communication*, vol. 93, p. 116135, 2021.

[79] H. Rashed, E. Mohamed, G. Sistu, V. R. Kumar, C. Eising, A. El-Sallab, and S. Yogamani, "Generalized object detection on fisheye cameras for autonomous driving: Dataset, representations and baseline," in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 2271–2279, 2021.

[80] T. Li, G. Tong, H. Tang, B. Li, and B. Chen, "Fisheyedet: A self-study and contour-based object detector in fisheye images," *IEEE Access*, vol. 8, pp. 71739–71751, 2020.

[81] Q. N. Minh, B. L. Van, C. Nguyen, A. Le, and V. D. Nguyen, "ARPD: Anchor-free rotation-aware people detection using topview fisheye camera," in *2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, IEEE, nov 2021.

[82] T. Wang, C.-W. Chang, and Y.-S. Wu, "Template-based people detection using a single downward-viewing fisheye camera," *2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pp. 719–723, 2017.

[83] A.-T. Chiang and Y. Wang, "Human detection in fish-eye images using hog-based detectors over rotated windows," in *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2014.

[84] O. Krams and N. Kiryati, "People detection in top-view fisheye imaging," in *14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017.

[85] R. Seidel, A. Apitzsch, and G. Hirtz, "Improved person detection on omnidirectional images with non-maxima suppression," in *VISIGRAPP*, 2019.

[86] S. Li, M. Tezcan, P. Ishwar, and J. Konrad, "Supervised people counting using an overhead fisheye camera," *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2019.

[87] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, pp. 1231 – 1237, 2013.

[88] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015.

[89] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020.

[90] "Offline data augmentation - nvidia docs." https://docs.nvidia.com/tao/tao-toolkit/text/offline_data_augmentation.html. Accessed: August 23, 2023.

[91] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[92] A. Kuznetsova, H. Rom, N. Alldrin, J. R. R. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and V. Ferrari, "The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale," *International Journal of Computer Vision*, vol. abs/1811.00982, 2018.