Tampere University

Joram Puumala

MONITORING DANGEROUS GOODS ON ROADS USING COMPUTER VISION

M.Sc Thesis
Faculty of Information Tehcnology
Examiners: Assoc. Prof. Esa Rahtu
February 2023

ABSTRACT

Joram Puumala: Monitoring Dangerous Goods on Roads Using Computer Vision
M.Sc Thesis
Tampere University
Master's Degree Programme in Information Technology; Data Science and Machine Learning
February 2023

The transportation of dangerous goods on roads poses significant risks, as accidents involving hazardous materials can be deadly and devastating, especially in densely populated areas and confined spaces like tunnels. In the event of an accident, fast emergency response is crucial. Vehicles carrying dangerous goods must be marked with orange hazmat plates, one attached to the front and another to the back of the vehicle. This establishes a universal way to recognize these trucks within ADR member states. This thesis aims to determine if it is possible to use computer vision to automatically monitor dangerous goods in an efficient manner based on the visual information provided by the hazmat plates.

This research began with a review of relevant literature and background information. Then, a dataset was created for training deep learning-based object detectors to identify and locate trucks and hazmat plates in images. A method for classifying trucks as carrying dangerous goods was also developed. Finally, the effectiveness of the method was tested on a set of videos of highway traffic that were collected manually.

The proposed approach for dangerous goods detection and monitoring in this thesis proved to be highly accurate. The method was evaluated using a set of highway traffic videos with varying levels of complexity, including scenarios where false positives could occur. The proposed method was able to identify all trucks carrying dangerous goods and avoid false positives. The method also proved to be highly efficient, as it could process 26 frames per second on a modern edge device.

Keywords: Computer Vision, Object Detection, Object Tracking, Dangerous Goods, Deep Learning

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

PREFACE

I am extremely grateful to Markus Ahonen, who has consistently supported me throughout the years and provided me with the opportunity to pursue my passion by tackling challenging real-world issues. This thesis topic, which he assisted me in selecting, is yet another example of this.

I would also like to extend my deepest gratitude to Assoc. Prof. Esa Rahtu for his official supervision on this thesis at Tampere University. I am deeply appreciative of his guidance and the enlightening feedback received throughout the course of this project.

In closing, I would like to express my deepest appreciation to my family and friends for their constant support throughout the entire process. I am particularly grateful to my fiancée Anna, whose invaluable assistance and counsel were instrumental in the completion of this thesis.

In Tampere, 26th February 2023

Joram Puumala

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

ADR	Agreement of 30 September 1957 concerning the International Carriage of Dangerous Goods by Road
AP	Average Precision
API	Application Programming Interface
CCTV	Closed-circuit television
CNN	Convolutional Neural Network
DGs	Dangerous Goods
FPS	Frames per second
GPS	Global Positioning System
GPU	Graphics Processing Unit
IoU	Intersection over Union
mAP	Mean Average Precision
MOT	Multiple Object Tracking
NMS	Non-Maximum Suppression
SORT	Simple Online Real-time Tracker
SSD	Single Shot MultiBox Detector - One-stage object detection algorithm
YOLO	You Only Look Once - One-stage object detection algorithm

1. INTRODUCTION

The transportation of dangerous goods (DGs) is essential to the modern world as it moves gases, explosives, liquids, and many other items from refineries to their final destinations. The most commonly transported substances are petroleum products such as gasoline, which powers internal combustion engine cars. These products are usually transported on roads [1] by heavy duty trucks, which means they share the roads with other vehicles and thus pose a major risk for accidents. Accidents involving DGs cause most destruction in densely populated areas, near critical infrastructure, and in confined spaces such as road tunnels, where toxic gases, fires, and explosives pose a significant threat to human lives [2, 3]. Such accidents can also cause destruction to infrastructure and lead to environmental disasters. While these types of accidents are not common, statistics show that 5203 lives were lost between 2006 and 2017 in China due to incidents involving hazardous materials [4]. One of the deadliest such accidents occurred in Yanhou in 2014 when two trucks carrying methanol crashed in a road tunnel, causing a fire that resulted in 40 deaths and 12 injuries [5].

As more and more products are being delivered globally, there has been a continual increase in road traffic in tunnels, particularly for heavy duty vehicles. This increase in traffic also increases the risk of accidents. However, a complete ban on the transportation of dangerous goods in tunnels could have unintended economic consequences and may lead drivers to choose potentially more dangerous routes through densely populated areas or near critical and vulnerable services such as hospitals [2]. Therefore, it is necessary to develop new safety methods and tools to assist first responders.

It may be unrealistic to expect accidents to be completely eliminated as long as there are human drivers on the roads. The unpredictable nature of humans is a major cause of accidents, and while research has been conducted to identify and mitigate human and environmental factors that increase the risk of accidents, there is still much that cannot be controlled [6]. Responding effectively to incidents, especially those involving dangerous goods, is important for improving safety on the roads. Previous research has focused on the response to incidents involving dangerous goods [7] and the use of deep learning methods for detecting general accidents in tunnels [8]. However, little research has been done on the automatic detection of dangerous goods on roads. In the past, attempts at this have produced limited, low-accuracy results and did not address the issue of in-

ference speed [9]. However, with the advancement of deep learning in object detection and improved computer hardware, it is worth revisiting the concept of automated tools for detecting dangerous goods on roads

The goal of this thesis is to test different object detection methods for truck and hazmat plate detection near tunnel entrances and find a robust way to associate the detected hazmat plates with detected trucks. The objective is to find an accurate and efficient detector which can be combined with an object tracker for real-time dangerous goods monitoring, optimally on a modern edge device. However, the identification of different substances based on the numbers written on hazmat plates is not included in this work. In the absence of any suitable datasets for the research outlined in this thesis, the methods for constructing image datasets are also examined, leading to the development of a new object detection dataset from scratch for the purpose of this research.

The thesis is structured as follows: The background information on the topic can be found in Chapter 2, followed by an explanation of the methodologies used in dataset construction and experiments in Chapter 3. The process of building object detection and object tracking datasets is outlined in Chapters 4 and 5. The experiments, including their results, are discussed in Chapter 6. The final chapter, 7, includes a conclusion and final thoughts.

2. BACKGROUND

This chapter provides the necessary background information for the rest of the thesis. It begins by discussing the current state of object detection and object tracking. Next, it covers the process of building datasets from scratch. The chapter concludes with a discussion of the transportation of dangerous goods by road in ADR member states.

2.1 Object Detection

Object detection is one of the most fundamental, important and challenging problems in computer vision domain. The task involves identifying the presence and location of objects in image. It has gained popularity in recent years due to advances in deep convolutional neural networks, availability of powerful computing platforms such as graphics processing units (GPUs), potential for a wide range of interesting applications and other factors. In the past, object detection focused mainly on identifying specific types of objects, such as faces or pedestrians, but it has since expanded to encompass general object detection. Today it plays an important role in applications such as surveillance systems, autonomous driving (Figure 2.1) and industrial monitoring.

Object detection can be difficult due to various challenges such as occlusion, lighting changes, camera parameters, motion and different weather conditions. The task can also be made more difficult by the wide range of variations within a single object class, including differences in pose, texture, color, and size. For example, accurately distinguishing between a leopard and a cheetah can be difficult, and this becomes even harder in less than ideal conditions. In addition to improving the accuracy of object detection algorithms, there is also a need to improve their efficiency. This is partly because the computational complexity increases as the number of categories and objects in an image grows, and real-time performance is often required for applications running on mobile and edge devices.

Object detection is more complex than image classification because it involves not only identifying what objects are present in an image, but also determining their location within the image. This added complexity and the need for large amounts of annotated data for training can make object detection challenging to scale. While annotating images for image classification is typically fast and easy, annotating images for object detection

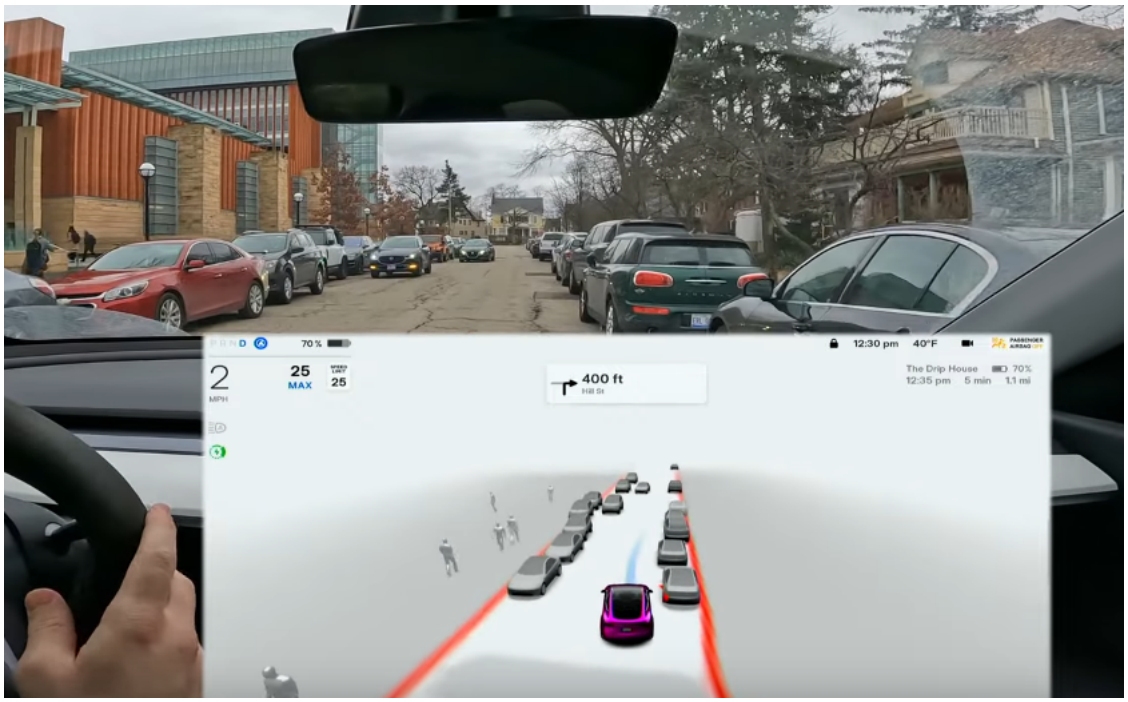


Figure 2.1. *Object detection plays a big role in autonomous driving. Tesla's Full Self-Driving (FSD) Beta program maps the vehicle's surroundings into a vector space from multi-camera input, where vehicle's path and actions are predicted. Image courtesy of Chris McBob. Used with permission. The original work is available in [10].*

requires marking the location of each object of interest with a bounding box (Figure 2.2), which can be time-consuming and labor-intensive. This is especially true when there are a large number of objects in an image or a large number of object categories to consider. Research is ongoing in methods and tools for assisted labeling to help make the annotation process more efficient [11]. Some commercial tools are already available [12].

Recent object detectors can be broadly classified into two categories: two-stage detectors and one-stage detectors. In the past, two-stage detectors have generally had higher accuracy on various benchmark datasets, but they tend to be slower. One-stage detectors, on the other hand, are faster due to their efficient architecture, but have not traditionally achieved the same level of accuracy as two-stage detectors. However, recent advances in one-stage detector design are closing this accuracy gap [13].

Two-stage detectors have two separate stages for detecting objects in an image. The first stage generates a set of candidate bounding boxes, and the second stage processes these candidates to produce the final set of bounding boxes for the detections. Some of the earliest two-stage detectors, such as R-CNN [14] and SPPNet [15] were extremely slow and also hard to optimize, because the two stages had to be trained separately. Fast RCNN [16] introduced a unified training process that allowed simultaneous training of a softmax classifier and a class specific regressor, leading to significantly faster train-

a) b)

Figure 2.2. *Bounding boxes are often used to show the location of objects in images. In some cases, multiple bounding boxes may overlap around the same object, as shown in a) above. This can occur when there are multiple good predictions for the object's location. To address this issue, a post-processing step called Non-Maximum Suppression (NMS) is often used to eliminate overlapping bounding boxes and ensure that each object is represented by a single bounding box. The result of this process is shown in b), where unnecessary bounding boxes have been removed.*

ing and inference speeds. Later, Faster R-CNN [17] introduced innovations such as the Region Proposal Network (RPN) that further improved the efficiency and accuracy of the two-stage detectors. Mask R-CNN [18] extended Faster RCNN to enable pixel-level object instance segmentation. The improvements introduced by Faster R-CNN resulted in state-of-the-art accuracy on PASCAL VOC 2007 [19], while running at over 10 times the speed of the earlier two-stage detectors. Despite these advances, two-stage detectors still struggle to achieve real-time inference speeds while maintaining high accuracy.

One-stage object detectors have a single stage that directly predicts bounding boxes and class probabilities from an image in a single pass using a convolutional neural network (CNN). Unlike two-stage detectors, one-stage detectors do not have a separate region proposal network for generating candidate bounding boxes. Instead, bounding boxes are commonly regressed from anchor boxes placed at multiple scales within a grid of cells on the feature maps of the CNN, as illustrated by Figure 2.3 [20, 21]. Anchor boxes are predefined bounding boxes that serve as a starting point for the regression process. In order to improve the accuracy of the detections, grids at different scales are used, with smaller grids being better at detecting small objects and larger grids being better at detecting larger objects. For example, the YOLOv4 detector uses grids at three different scales (52x52, 26x26 and 13x13) with 3 anchor boxes per grid, resulting in a total of 10920 predicted bounding boxes. While anchor-based one-stage detectors are popular, there are also anchor-free methods, such as CenterNet [22], which uses keypoint triplets for detection.

The feature extractor portion of an object detector is often referred to as the backbone

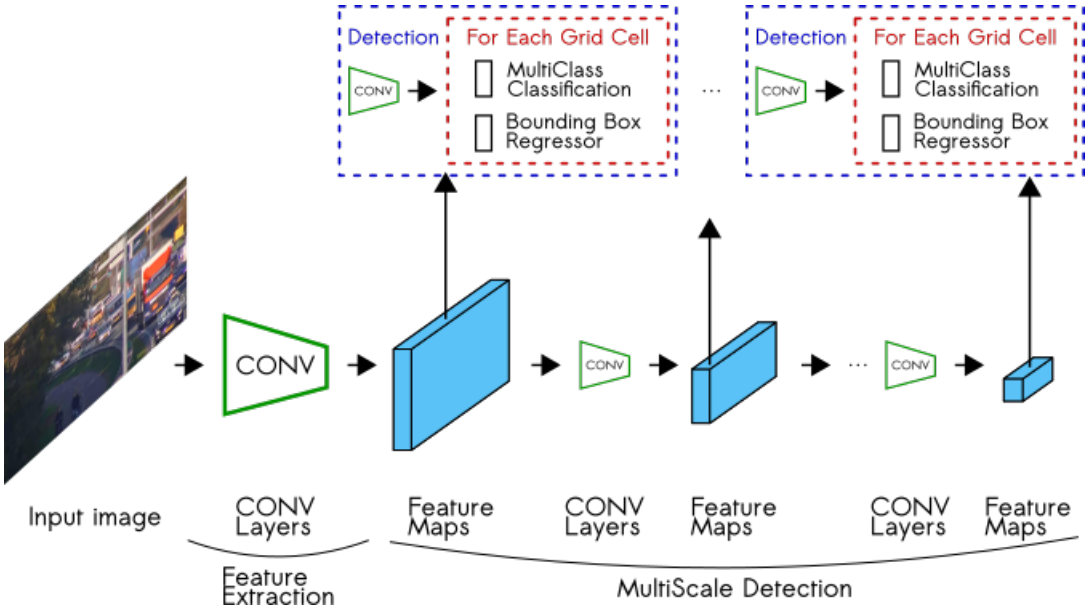


Figure 2.3. *A typical basic architecture of one-stage object detectors such as YOLOv4 and SSD. In this architecture, features are extracted from the input image using convolutional layers, and the resulting feature maps at different scales are used to detect objects and output bounding boxes and class labels. The entire sequence of operations is implemented as a single, unified fully convolutional neural network.*

network, while the detection portion is called the head. Some detectors may also include an intermediate feature extractor called the "neck" between the backbone and the head. The distinct naming comes from the common practice of pre-training the feature extractor (backbone) on a large image dataset and then attaching the detection part (head) and fine-tuning the model [15, 17, 20, 23]. The feature extractor is typically a classification network such as ResNet [24] or VGG [25], with some of the final layers removed. This modular design allows for easy experimentation with different object detectors by changing the backbone network, which can affect the detector's accuracy and speed. For example, using a ResNet 101 backbone for Faster R-CNN may result in a more accurate detector than using a MobileNet [26] backbone, but the latter would likely be faster and more suitable for platforms with limited computing power.

Many modern object detection methods generate a large number of predictions per image, even after thresholding by detection confidence. To address this issue, post-processing algorithms such as Non-Maximum Suppression (NMS) are often used [15, 17, 20, 23] to remove overlapping predictions and ensure that each object is represented by only one prediction, as shown in Figure 2.2. NMS works by selecting the highest confidence detection, calculating the Intersection over Union (IoU) score between this detection and all other detections, and then removing any detections that receive an IoU score higher than a given threshold. If there are still detections remaining, the process is repeated until all overlapping detections have been removed.

2.2 Object Detection Evaluation

Evaluating object detection algorithms is slightly more complex than evaluating image classification algorithms, which can be evaluated by simply counting the number of correct classifications and dividing by the total number of images. In object detection, the predicted bounding boxes are compared to the ground truth bounding boxes to determine how well the model is performing. When a predicted bounding box and its class match a ground truth bounding box and class, it is considered a true positive. To determine if there is a match, the Intersection over Union (IoU) between the bounding boxes is often calculated and thresholded. If a bounding box does not match any ground truth bounding boxes or the class is predicted incorrectly, it is considered a false positive. Any ground truth bounding boxes that do not have a corresponding match are considered false negatives.

The predicted bounding boxes are usually ranked according to their confidence scores and the results are presented in terms of precision and recall. In some cases, the values are interpolated, meaning that they are grouped together in steps. For example, 11-point interpolation would only consider the first decimal place of the confidence value when grouping the bounding boxes.

Precision is the positive predictive value. It shows the proportion of positive predictions to negative predictions. The formula is written as,

$$Precision = \frac{True\ positives}{True\ positives + False\ positives}. \tag{2.1}$$

Recall on the other hand, is the true positive rate. It tells how well the model is able to detect the true objects. The formula is written as,

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives}. \tag{2.2}$$

The calculations for precision and recall result in lists of values at different confidence thresholds. These values can be plotted on a graph, with the x-axis representing recall and the y-axis representing precision, to create a precision-recall curve (Figure 2.4). The area under this curve is often calculated and referred to as the average precision (AP) for a given class. The mean average precision (mAP) is obtained by averaging the class-wise AP values, providing an overall measure of the performance of the object detection model. The mAP is a commonly used metric for comparing object detection models [17, 20, 21].

As previously mentioned, the Intersection over Union (IoU) is typically used to match predicted bounding boxes to ground truth bounding boxes. High IoU threshold values require

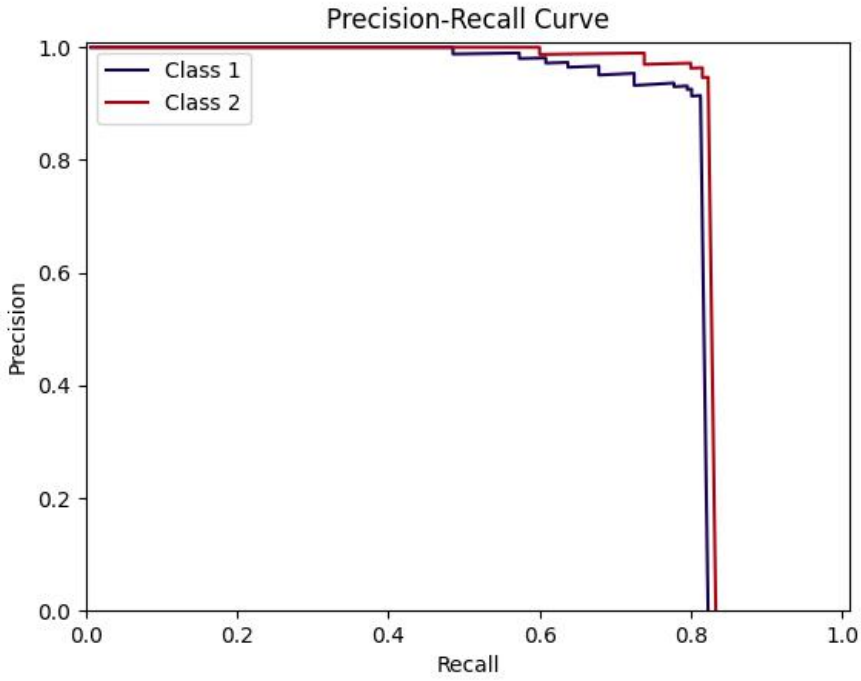


Figure 2.4. *Precision-Recall curve illustrated. Precision and recall values are plotted class wise. Calculating area under each curve results in AP for each class. Average of all the AP values of all classes is known as mAP.*

the predicted bounding box coordinates to closely match the ground truth, which rewards detectors that are able to accurately localize objects. The appropriate IoU threshold for a given object detection problem depends on the dataset and how it was constructed. For example, the Pascal VOC dataset uses a relatively low threshold of 0.5 due to the subjectivity of determining the boundaries of certain objects, such as humans, when labeling the data [19]. In contrast, the MS COCO challenge specifies that the mAP value should be calculated at different thresholds ranging from 0.5 to 0.95 with increments of 0.05 and averaged to obtain the final result [27].

2.3 Multiple Object Tracking

Multiple Object Tracking (MOT) is a computer vision technique that aims to identify and follow multiple objects in a video. This is done by analyzing the video input and assigning unique IDs to each detected object. Unlike the task of object detection, MOT focuses on keeping track of the detected objects. MOT algorithms do not have prior knowledge of the number or appearance of objects, unlike Single Object Tracking algorithms (SOT), so a detection step is required to locate the objects before they can be tracked. This process, known as tracking-by-detection, is the standard approach for MOT. The MOT task is an important area of research in computer vision, and robust MOT algorithms are necessary for applications such as autonomous driving, where it is important to track objects like

vehicles and pedestrians in order to predict their movement. Tracking multiple objects simultaneously can be challenging due to factors such as occlusions, pose changes, object interactions, and a moving camera [28]. MOT algorithms also need to be able to re-identify objects that are lost and reappear. The object detection step, which guides the tracking process in the tracking-by-detection paradigm, has a significant impact on the tracking results [29].

MOT algorithms can be divided into two categories: batch (offline) and online methods. Batch methods are able to use information from future frames to make tracking predictions in the present, while online methods can only use past and present frames. Systems that require real-time processing are limited to using online methods, but not all online methods are able to run in real-time. Batch algorithms often perform better than online algorithms in evaluation metrics, especially when there are a high number of fragmentations, which occur when object tracking is momentarily interrupted due to a missing detection. However, the gap in performance between batch and online methods has been closing in recent years [29].

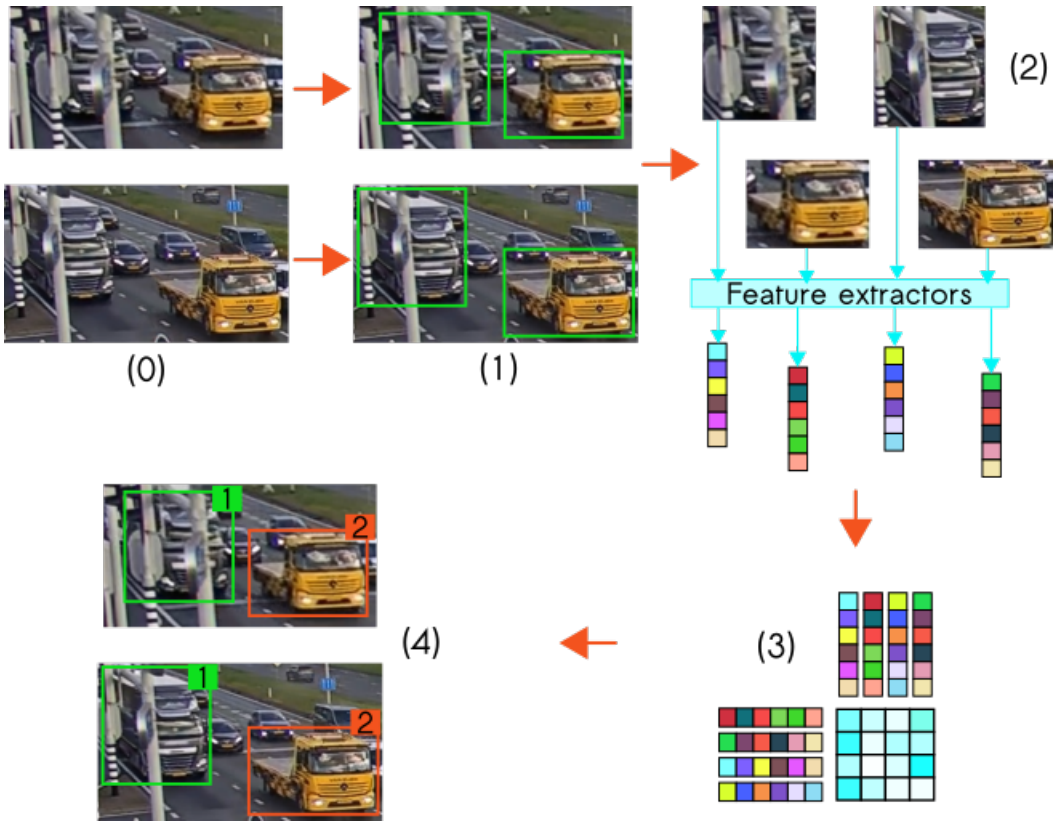


Figure 2.5. *The typical process of a MOT algorithm can be broken down into the following steps: (0) the algorithm receives raw input frames, (1) an object detector analyzes the frames and outputs bounding boxes for the objects of interest, (2) feature extractors compute various features (such as visual and motion features) for each detected object, (3) the probability of two objects belonging to the same target is calculated, and finally (4) each target is given a unique ID.*

There are many approaches to MOT, but most algorithms follow some or all of the following steps: (1) detection, (2) feature extraction, (3) affinity calculation, and (4) association. The different stages are illustrated in Figure 2.5. In the detection stage, the algorithm identifies objects of interest and determines their bounding box coordinates. During the feature extraction stage, the algorithm analyzes the detected objects for appearance and/or motion, and may also predict the future positions of the tracked objects. In the affinity stage, the learned features and motion predictions are used to compute similarities or distances with the detected objects or tracklets. Finally, in the association stage, the results of the similarity or distance calculations are used to associate the detected objects with existing ones [30, 31].

Deep learning has become an important part of modern object detection and tracking algorithms. In addition to the detection stage, deep learning can be used in other stages of the process, such as in the feature extraction phase, as seen in approaches like DeepSORT [30] and those that use Siamese networks [32, 33, 34]. Deep learning has also been applied to later stages of the process using recurrent neural networks (RNNs) and long short term memory (LSTM) cells [35, 36]. However, even with the recent success of deep learning in object tracking, hand-crafted distance metrics are still frequently used in combination with neural networks, and there are still significant MOT algorithms that do not use deep learning techniques. An example of a non-deep learning based online object tracking algorithm is SORT, which relies on the Kalman filter [37] and the Hungarian algorithm [38] to keep track of objects [31].

2.4 Multiple Object Tracking Evaluation

Datasets provided by MOTChallenge [39] are a popular choice for benchmarking object trackers. These datasets are concerned with pedestrian tracking. Another popular dataset for benchmarking is KITTI [40], which provides a challenge to track pedestrians and vehicles. These benchmarking datasets supply all detections to the object trackers, to make it a fair game by only comparing the tracking ability. The performance is evaluated by a group of metrics described below.

MOTChallenge datasets [39] are widely used for evaluating the performance of object tracking algorithms. These datasets focus on tracking pedestrians. Another commonly used benchmarking dataset is KITTI [40], which includes challenges for tracking pedestrians and vehicles. In order to ensure a fair comparison, these datasets provide all of the detections to the object tracking algorithms being evaluated. The performance of these algorithms is measured using a set of metrics.

Two commonly used evaluation metrics for object tracking are MOTA (Multiple Object Tracking Accuracy) and MOTP (Multiple Object Tracking Precision). These scores are constructed from simpler metrics, and MOTA is defined as follows:

$$MOTA = 1 - \frac{FN + FP + IDSW}{GT} \in (-\infty, 1] \tag{2.3}$$

where FN represents the total number of false negatives, FP represents the total number of false positives, $IDSW$ represents the total number of ID switches, and GT represents the number of ground truth bounding boxes. MOTA can sometimes result in a negative score, which is why it is often expressed as a percentage. While MOTA evaluates the overall tracking performance in relation to the ground truths, MOTP focuses more on the quality of the detections. It is defined as follows:

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{2.4}$$

where $d_{t,i}$ represents the overlap between hypothesis i and the ground truth bounding box, and c_t represents the number of matches between the hypotheses and ground truths in frame i [41].

MOTA score has a major limitation: it takes into account the number of incorrect decisions made by a tracker, such as ID switches. In certain circumstances, this behavior may not be considered ideal. For instance, when using surveillance footage, one may prefer to reward a tracker that is able to follow an object for an extended period. To address this issue, the IDF1 metric was introduced. This metric globally matches trajectories instead of matching ground truth bounding boxes and detections frame by frame [42]. Other common metrics include the total number of fragmentations, mostly tracked trajectories, and mostly lost trajectories [39, 41]. Tracking speed is also often considered, but it can be difficult to compare as some methods include the detection phase while others do not.

2.5 Construction of Image Dataset for Object Detection

Constructing a dataset is a complex task that can be broken down into smaller subtasks. This process typically involves 3 major steps: collecting data, preprocessing the data, and labeling the data. Each of these steps is important for producing a useful dataset for training an object detection model.

The process of acquiring data for a dataset, in this case images, from the real world through various means and storing it in a storage system like a solid-state drive drive or cloud storage offered by a third party is called data collection [43]. This is a crucial aspect of constructing a dataset, and having a carefully planned strategy for data collection can make subsequent steps, such as data cleaning, easier. There are various methods for collecting image data, including manual, semi-automated, and automated techniques.

Manual data collection refers to the practice of manually triggering the capture of each

data point through human input. This can be time-consuming and costly, as it requires a lot of human effort and may take time away from other projects that require the expertise of field experts. For example, manually collecting information about the condition of curbs and quality of pavement markings on the side of the road is a slow process that typically involves workers walking or driving along the road and taking measurements with simple equipment [44]. While manual data collection can result in a more accurate initial dataset due to the ability of humans to selectively gather relevant data points, it may be difficult to scale up the dataset size using this method.

Semi-automated data collection refers to the use of methods that automate certain aspects of data collection that would otherwise be done manually. However, these techniques still require some human input and are not capable of carrying out long-term data collection without it. For example, in the context of collecting roadside data, some aspects of the process could be automated by using a car equipped with high-resolution cameras, GPS, and other sensors to capture a large amount of data at highway speeds [44]. While this approach may result in a dataset with a much larger number of data points compared to manual data collection, it may also contain more noise due to the difficulties of programming logic that can selectively gather data in the same way as human experts. However, cleaning the dataset of noise may be less time-consuming than manually collecting all of the data.

Automated data collection refers to the use of methods that can capture data over an extended period of time without requiring human input. This approach is typically cost-effective but may be challenging to use without introducing noise into the dataset. There are many ways to automate data collection, and the specific method used depends on the nature of the problem. The widespread use of the internet has made it possible to collect large datasets using web scraping techniques [19, 45]. However, automated data collection is not limited to web scraping and can also be achieved through other means. For example, researchers in [46] used a physical camera to automatically capture and upload images for further processing.

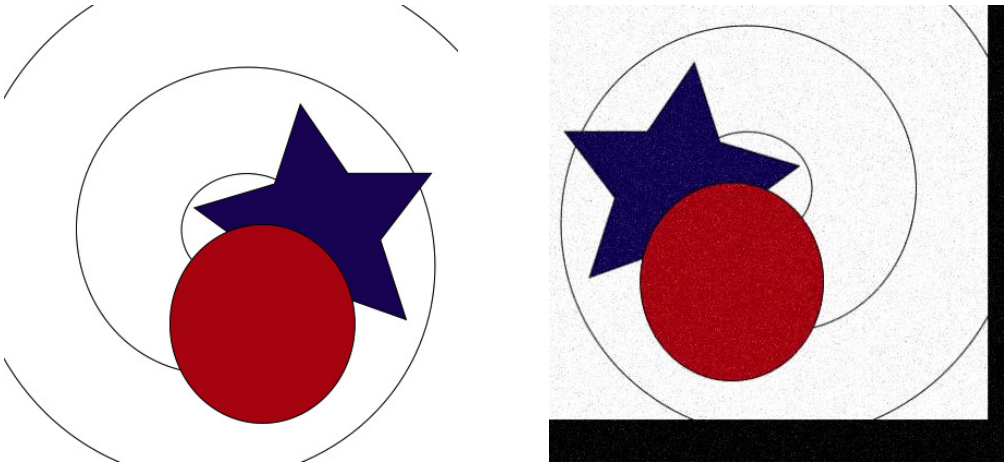
In practice, constructing a dataset often involves a combination of manual and automated techniques. If the available data is scarce and highly specific, more complex and fine-grained automated methods may not be practical to use until some data has been collected through manual means. This is often the case in closed manufacturing environments, where the data may be highly specific and not available for download from the internet or purchase in a market.

Data preprocessing is the process of preparing data for analysis by learning algorithms. It includes all the steps taken before the data analysis begins. Many datasets, particularly large ones, are constructed using automated or semi-automated methods, which can result in noisy data such as low-quality images [47]. Some large face datasets have

demonstrated that poor quality samples can make up as much as 30% of the data [48].

Cleaning a dataset of low-quality data is important to ensure its overall quality. This process often involves tedious manual labor, where humans inspect the images and remove those that do not meet the required quality standards. Some research has attempted to partially automate dataset cleaning [47], but current methods tend to be specific to a particular domain rather than a general solution.

Data cleaning can improve the quality of a dataset by removing low-quality data points, but further improvement is often possible by adding augmented data to the dataset. Data augmentation techniques, which transform existing training data into new data points through geometric and color transformations, the addition of noise, random erasing, and other variations, can be particularly useful for small datasets or when the data is very monotone. An example of an image transformation is shown in Figure 2.6. CNN-based object detectors may overfit and struggle to generalize well if the data is insufficient [49, 50, 51]. In recent years, generative adversarial networks (GANs) have become popular for use in data augmentation tasks, in addition to traditional image transformations [50]..



a) b)

Figure 2.6. *Original image a) is transformed into a new data point b) during data augmentation process. In this case, the original image is flipped horizontally, and then further translated along x and y axes and blurred with laplace noise.*

The goal of data augmentation is to add diversity to the dataset and minimize the distance between training and test data. By including translational invariances in the training data, the model can learn from a wider range of inputs and is less likely to overfit. Data augmentation can also be used to balance imbalanced datasets [50]. Well-established data augmentation methods have been shown to improve the performance of CNNs on unseen data, though no single approach is universally the best. The effectiveness of a particular technique depends on the specific dataset [49].

Data augmentation should be performed on a dataset after it has been cleaned, to prevent the proliferation of low-quality images as the dataset grows. Furthermore, data augmen-

tation is typically applied after the labeling process, so that the same modifications can be made to both the images and their corresponding bounding boxes. However, when using data augmentation for object detection, there are certain considerations and limitations that should be taken into account. For instance, flipping an image of a digit 6 vertically, can cause it to be mislabeled as a 9, if the label is not corrected [50]. Also, object detection labeling limits the amount of rotations that can be done, since rotations greater than 90 degrees at a time may cause the augmented bounding box to lose its optimal fit around the object, unless the object has a circular shape.

In addition to removing poor-quality data and increasing the size of the dataset through data augmentation techniques, the images may also be converted to a fixed shape at this stage, as convolutional neural networks typically require a fixed input size. Furthermore, during run time, it's common practice to normalize the data by subtracting the mean RGB values from the images. This value is calculated only on the training data, but the same value is subtracted from the test data as well [24, 25, 52].

Data labeling is the task of assigning relevant labels to specific data points within a dataset. When it comes to object detection, labels are assigned as unique numbers or words, and coordinates are used to indicate the location of the object within an image. Each data point is associated with coordinates that define the bounding box area of the object, usually represented by x and y values for the starting and ending points [19] or as center x and y values along with the width and height of the bounding boxes [27].

Data labeling for object detection is a tedious task, as it often requires a significant amount of manual labor [46]. Some researchers have tried to ease this process by developing algorithms that make the labeling process faster and more efficient. For example, by using machine learning to generate initial labels, which then need to be verified and corrected by human annotators. These methods have the potential to save a considerable amount of time, but their effectiveness can vary depending on the dataset [11]. Additionally, there are also commercial tools that are available that assist with the labeling process [12].

The task of labeling data for object detection is typically a collaborative effort and not typically done by one individual alone, particularly when working with large datasets which are becoming more common. This often results in multiple people working on the labeling process of a single dataset. Ensuring the quality of the labeling process is crucial for building successful object detection models, as poor quality data leads to less effective models. In object detection, it's especially important to accurately label all instances of interest and consistently draw tight bounding boxes around them. Ensuring the quality of the data labeling process often involves proper training of annotators and providing clear instructions. As the dataset grows and more people are involved in the labeling process, it becomes increasingly important to ensure the quality of the labeling process to avoid subjectivity for certain object boundaries, which is common in crowdsourcing based data

labeling [53].

Some large technology companies, such as Tesla, have their own dedicated in-house teams for data labeling [54] but many companies may not have the resources to do this internally. Crowdsourcing has become a common solution for such companies, where a large group of dispersed individuals annotate the data through platforms like Amazon Mechanical Turk. However, maintaining the quality of the labels can be more challenging when the process is outsourced [53]. Additionally, crowdsourcing may not always be a feasible option, as the data may be confidential and cannot be shared with external individuals [11], or the labeling process may require specialized knowledge or expertise [50].

2.6 Transportation of Dangerous Goods by Road in ADR member states

Regulations for transporting hazardous materials by road may have some variations among countries. However, countries that are part of the ADR adhere to a consistent rulebook. ADR is an agreement that governs the international transportation of dangerous goods by road. Most importantly, the agreement stipulates that hazardous materials can generally be transported internationally by road in wheeled vehicles, with certain exceptions for extremely hazardous materials, provided that the vehicles and equipment comply with the specified guidelines and are properly marked. As of May 2022, ADR has 53 member countries [55]. The ADR member states are visualized in Figure 2.7.

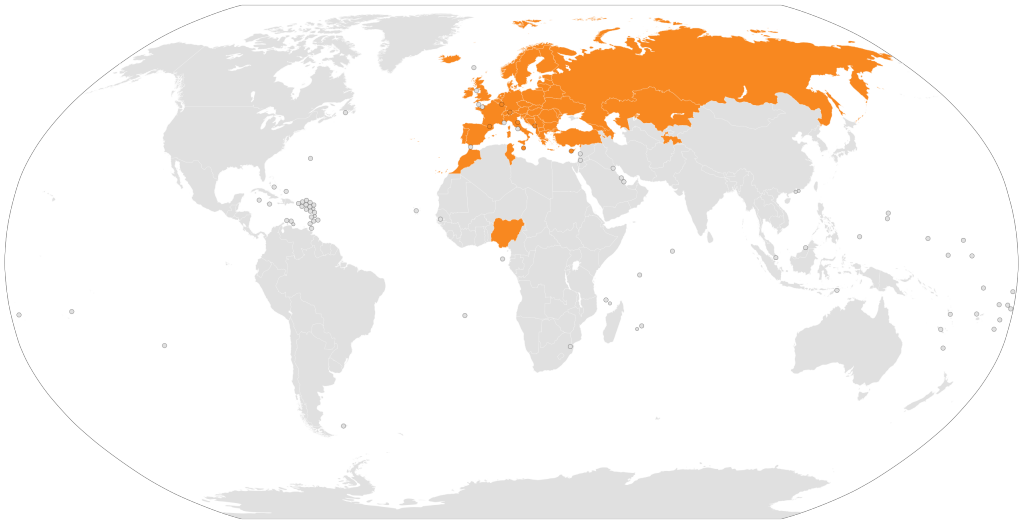


Figure 2.7. *Map of the ADR member states. Countries painted orange represent the member states of ADR.*

The ADR sets out specific placards to represent different types of hazardous materials (hazmat). These placards provide detailed information on the hazardous materials being transported. Additionally, ADR also requires that every vehicle transporting dangerous

goods by road display an orange-colored plate marking on the front and rear of the vehicle. The background of the plate is always orange, and its borders are always black. Some plates have two numbers written in black, separated by a horizontal black line in the middle, as Figure 2.8 shows. The numbers on the plate indicate the hazard identification number (2-3 digits on the top, sometimes preceded by letter X) and UN number (4 digits at the bottom). Alternatively, the plate can also be a solid orange rectangle with black borders [55].

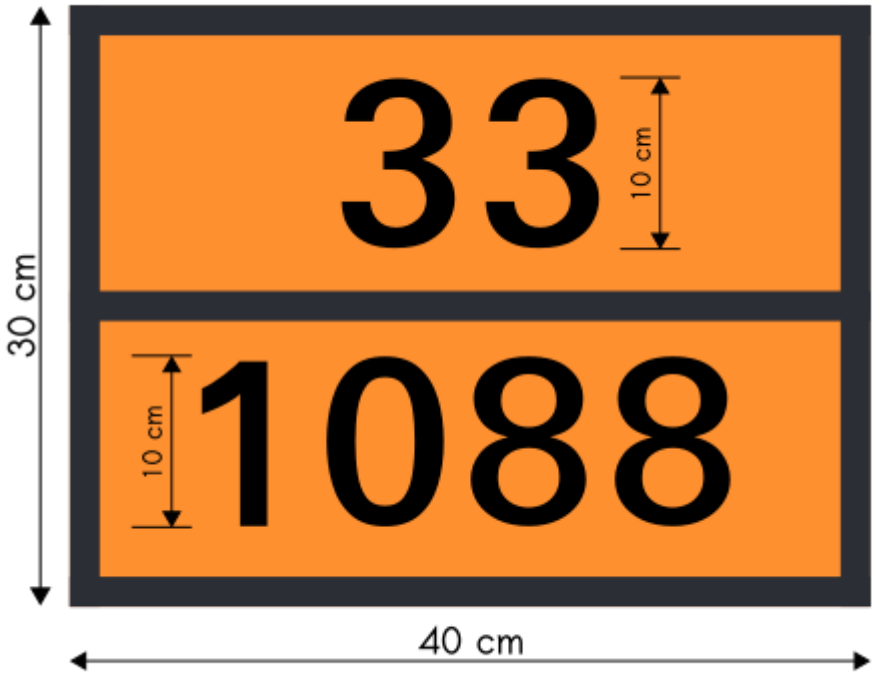


Figure 2.8. *The orange-colored plates have specific dimensions. The upper number on the plate indicates the hazard identification number and the lower number represents the UN number. It is also common to see a hazmat plate without the numbers and the middle horizontal line.*

The hazard identification number on the plate identifies the type of hazard posed by the materials being transported, such as radioactivity or an oxidizing effect. The types are represented by numbers from 2 to 9, a total of 8 types. A double digit number indicates that the particular hazard is intensified, unless the last number is 0, which indicates there is no intensified effect. The presence of the letter X before the digits warns that the substance has a dangerous reaction with water. Additionally, some digit combinations carry specific connotations and implications [55].

The UN number, or United Nations number, is a 4 digit identifier for hazardous materials. It should be noted that, the UN number alone does not provide information about the general class of the hazardous material [55].

The orange-colored hazmat plates must be reflectorized, weather-resistant and able to remain affixed to its mount for a minimum of 15 minutes in the event of a fire. According to ADR regulation, the length of the base of the plate must be 40cm and the height 30cm, except in cases where the surface area is too small to attach the orange plates. In such cases, the minimum length of the base is 30cm and the height must be at least 12 cm [55].

The orange-colored hazmat plates are a universal way to recognize vehicles transporting dangerous goods on roads within ADR member states, which make it theoretically possible to automate monitoring of such cargo. However, there are certain challenges that arise for applications attempting this task. For example, the plates are small and it can be difficult to get a good recording angle. Low light conditions also pose a significant challenge, as there are no requirements for lighting near the plates. These challenges, along with other factors, need to be considered when developing such automated monitoring systems.

3. METHODS

This chapter provides an overview of the methodology used to build a dataset from scratch for truck and hazmat plate detection and proposes a method for detecting dangerous goods on roads. The chapter begins by discussing the process of dataset construction, and continues by presenting a method for associating truck and hazmat plate detections to form the final output. Finally, the chapter concludes with a discussion of the evaluation methods used to assess the performance of object detection models and the proposed approach.

3.1 Dataset Construction Methods

This thesis combines both manual and automated techniques for building object detection and object tracking datasets. Automated methods are mainly utilized during the data collection phase, where computer vision tools are employed to gather specific image data. The majority of the data is obtained through programming, by accessing still images or video feeds from online sources (road and traffic cameras). Data augmentation is also done using automated transformation functions. However, other stages of dataset construction, such as data cleaning and object detection labeling, rely heavily on manual labor.

3.2 Proposed Method for Detecting Dangerous Goods on Roads

According to ADR regulations, vehicles transporting hazardous materials in member states must display orange hazmat plates on the front and back of the vehicle. This work proposes a simple three-step method for identifying and tracking these vehicles.

In the first step, an object detector localizes trucks and hazmat plates in a frame, returning their bounding box coordinates. In the second step, the algorithm loops over the bounding boxes of the detected hazmat plates, and checks whether the coordinates are completely within any of the bounding boxes of the detected trucks. The algorithm associates each hazmat plate with the first matching truck and stops by returning the coordinates of the truck's bounding box. In the third and final step, these coordinates are fed into an object tracker, which keeps track of the dangerous goods. While the detection of the dangerous

goods is crucial, the tracking step makes it possible to use this information for a variety of practical applications, such as counting the number of objects.

The proposed method offers a flexible and robust framework for detecting and tracking dangerous goods. The efficiency of the first step heavily relies on the object detector chosen for the task. One-stage detectors such as YOLO and SSD can achieve real-time processing speeds, while two-stage detectors like Faster RCNN may not reach those levels. In theory, dangerous goods could also be tracked by the hazmat plates alone, but the association step makes this approach more robust against false positives. The downside is that the computational complexity of the association step is quadratic at worst. However, in practice, the likelihood of encountering a line of trucks carrying dangerous goods is minuscule.

3.3 Evaluation Methods

Object detection models in this work are evaluated with mean average precision (mAP), which is a common metric used for measuring the overall performance and creating a point of comparison for the models. This work computes the mAP of the models in two different ways. First, the object detection models are evaluated with an IoU threshold of 0.5, which is also how object detection models are evaluated in PASCAL VOC challenge [19]. The main comparisons are done this way. Second, the models are further evaluated by computing the mAP at different thresholds between 0.5 and 0.95 with 0.05 increments, their average results in the final mAP value. This follows the guidelines of MS COCO dataset [27], and rewards detectors with better localization capabilities. The final application is evaluated with arithmetic mean of precision and recall values for dangerous goods. Because the final application is evaluated on video data, true positives, false positives and false negatives are counted differently than they would be in individual images. The details of the counting process are laid out in algorithm 1.

The algorithm reads a sequence of frames and returns the number of true positives, false positives and false negatives. Each predicted bounding box receives a unique identifier (ID) by the object tracker. True positives are the predictions that match their respective ground truth bounding boxes and keep their assigned ID throughout their existence. False positives, on the other hand, are the predicted bounding boxes that do not match any of the ground truth bounding boxes or their IDs change during their existence. False negatives are all the ground truth bounding boxes that were never matched. The algorithm thus makes sure that the values are counted globally and not in individual frames.

In theory, the proposed method in the final application does not require an object detector to achieve a perfect mAP score to work optimally. Essentially, the detector only has to be able to detect the truck carrying dangerous goods in one frame. Traffic cameras are typically positioned in a way that allows them to capture vehicles for several seconds

Algorithm 1: Detection Verification. The algorithm takes a sequence of frames and ground truth annotations as input, runs inference on every frame in order, and outputs numbers of true positive-, false positive- and false negative detections.

Data: Detections and ground truths with their IDs over all consecutive frames
Result: true positives, false positives, false negatives

```
begin
    tp, fp, fn ⟵ (0, 0, 0)
    existing_pairs ⟵ {}
    for frame in all_frames:
        detections, detections_ids ⟵ inference(frame)
        for i in 0..n detections:
            highest_iou, highest_id ⟵ find_highest_iou(detections_i, gts)
            if highest_iou > threshold:
                if detection_ids_i not in existing_pairs:
                    existing_pairs ⟵ (detection_ids_i, gts_ids_highest_id)
                if existing_pairs_detection_ids_i == gts_ids_highest_id:
                    tp ⟵ tp + 1
                else:
                    fp ⟵ fp + 1
                gts.pop(highest_id)
                gts_ids.pop(highest_id)
            else:
                fp ⟵ fp + 1
        fn ⟵ fn + length(gts)
```

as they pass by, even at highway speeds. If the camera records at a rate of 25 FPS, the object detector would likely have more than 100 frames to identify the vehicle. More importantly, the tracker needs to be able to follow the detected objects when there are missing detections in some frames.

The performance of the final application is evaluated based on precision and recall values, which are combined to compute the arithmetic mean value. The key question is how high the application's score can be, and at what minimum confidence threshold this is achieved. This can be formally expressed as,

$$threshold = argmax(\{f(t_1), f(t_2), ..., f(t_{100}\}))) \tag{3.1}$$

where,

$$f(t) = \frac{Precision_t + Recall_t}{2} = \frac{\frac{tp_t}{tp_t+fp_t} + \frac{tp_t}{tp_t+fn_t}}{2}. \tag{3.2}$$

where tp is the number of true positives, fp represents the number of false positives and fn denotes the number of false negatives respectively. The performance of the

application is maximized at the confidence threshold calculated using equation 3.1, and the score (between 0 and 1) is obtained using equation 2, with the threshold as input.

4. CONSTRUCTION OF A DANGEROUS GOODS DATASET FOR OBJECT DETECTION

As the research topic of this thesis falls into a niche area and there is a lack of prior research, there were no existing datasets to support the investigation. As a result, a dataset of trucks and hazmat plates was created as part of the research. This chapter will detail the methods used to collect the data, the process of constructing the dataset, and the characteristics of the final dataset.

4.1 Data Collection

The majority of the image data used in this study was obtained by recording video footage from road cameras in Finland and the Netherlands. Additional data was gathered from other sources to minimize false positives and create a separate test dataset. The data collection was primarily done using automated methods, utilizing Python programming and various data sources available on the internet. The data sources were selected based on their licenses, to allow for commercial use in the future.

Traffic Management Finland, which is owned by the Finnish government, offers a free API to access images captured by Finland's road cameras [56]. The general public has access to images captured approximately every 10 minutes. There are around 1900 functional cameras, of which around 1000 are of decent quality. These cameras capture trucks from different angles, distances, and weather conditions, as shown in Figure 4.1. However, the images do not often feature trucks transporting dangerous goods, and when they do, the trucks are not typically captured at close enough proximity, making it difficult to distinguish the orange plates.

Figure 4.1. *The images obtained from road cameras in Finland provide a diverse dataset of trucks captured from different angles, with varying distances and different weather conditions.*

Approximately half of the road cameras were removed from the data collection process due to poor image quality. The images from these cameras did not capture the features of the hazmat plates, making them hard to distinguish, even in daylight. The manual camera selection process helped to reduce the total number of images downloaded by around half. The remaining images were retrieved by a custom Python script utilizing the Traffic Management Finland API. The script collected around 500 000 images over a 5 day period.

The data collected in the Netherlands was provided by road cameras located in the province of Gelderland. The cameras broadcast live streams of traffic in the province on the "Gelders Verkeer" (English: Gelderland Traffic) YouTube channel. The live video feed enabled capturing the same object at different locations and distances (Figure 4.2), which was particularly beneficial for hazmat plate data collection as it was difficult to find such data. This significantly accelerated the data collection process. In the province of Gelderland, there were more than 20 traffic cameras available. Additionally, each traffic camera was set up to record from various angles.

To ensure that only relevant images were captured during the data collection process, a custom python script was used to stream live videos from traffic cameras. This script was set to process every 25th frame, which not only reduced the number of almost identical images saved but also decreased the computational burden. However, even with this limitation, the script still captured a large amount of irrelevant images. To further refine the dataset, YOLOv3 with pre-trained weights on the COCO dataset was utilized to identify frames with trucks and only those were saved while discarding the rest. This process was carried out for 3 weeks and resulted in the collection of over 100 000 frames of images of trucks from various traffic camera locations.

Most of the data collected was obtained from road cameras in Finland and the Nether-

Figure 4.2. *Access to live video streams allows the capturing of the same object over multiple frames, which can significantly increase the speed of data collection. In the image, the same truck carrying dangerous goods is captured over multiple frames.*

lands. Some additional data was gathered in Finland by taking photographs with an iPhone 11 and from the website creativecommons.org. The main purpose of collecting this data was to minimize the occurrence of false positives, such as road signs, tanker containers, silos, and orange crates. However, some images of trucks with hazardous materials plates were also obtained, primarily on Finland's highways and at gas stations.

4.2 Preprocessing

During the data collection phase, approximately 600 000 images were obtained in total. The data from the Netherlands had already been filtered to contain only images of trucks. The images from Finland were then also processed to achieve the same result by utilizing the YOLOv3 application with pre-trained weights from the COCO dataset to retain only images of trucks. This preprocessing step resulted in a decrease of images captured in Finland from 500 000 to approximately 10 000.

The goal of the data collection process was to gather images of both trucks and hazmat plates, but the primary focus was on hazmat plates as they were not as prevalent as heavy duty trucks. Additionally, it was observed that in every instance where hazmat plates were present, they were attached to trucks. Thus, any time images of hazmat plates were collected, images of trucks were also obtained. To address concerns about class imbalances, all images collected in the Netherlands that did not have hazmat plates were manually removed. As the data was in a sequential order, this process turned out to be efficient and easy. This filtering process resulted in a reduction of the number of images from 100 000 to 1004.

The same filtering process was ultimately applied to the images collected in Finland, but with one exception. Images taken by road cameras in Finland displayed a greater diversity of weather conditions, such as heavy snow or rain, so those were retained along with the

images of hazmat plates. However, this time, the manual process was notably slower and more cumbersome, as the images were not in a sequential order. This step resulted in a decrease of the number of images from 10 000 to 475.

4.3 Labeling

The images collected were labeled using LabelImg, an open-source software designed for object detection data annotation. The software enables users to draw bounding boxes around desired object classes and save the information in YOLO, PASCAL VOC, and CreateML formats. Other labeling tools were also considered, including those with auto-labeling capabilities. However, the bounding boxes predicted by assisted labeling software tools often required manual adjustments. In practice, this led to a labeling process that felt more tedious and slow than drawing the bounding boxes manually from the start.

The labeling process was completed by outlining trucks and hazmat plates with bounding boxes, which suggests that the object detection method identifies these objects individually rather than specifically searching for dangerous cargo. This strategy was adopted as it was necessary for the proposed method, and there were concerns about the limited size of the dataset for end-to-end detection, which could increase the risk of false positives and negatives. However, these were only speculations and needed to be verified. Therefore, the labeled dataset, which had 2 classes, was converted into a separate dataset with a single class, specifically dangerous cargo. The transformation was quick and easy as it could be fully automated using Python.

4.4 Data Augmentation

The presence of heavy-duty trucks carrying hazardous materials on the roads is relatively rare in comparison to other types of trucks and personal vehicles. Collecting data for these types of trucks can be a time-consuming task, particularly without access to a large network of live traffic cameras and substantial computing resources. Despite the effort put into this project, the total number of images collected for the training dataset (1562) was relatively small by today's standards. To address this, a data augmentation pipeline was developed to expand the existing training dataset. This pipeline was applied after the labeling process, which resulted in significant time savings as the bounding boxes could also be augmented. However, a trade-off of this approach was that the images could only be rotated in 90-degree increments to maintain the optimal fit of the bounding boxes around the objects.

The data augmentation pipeline for the dataset was primarily created using the imgaug library in Python. The pipeline reads one image at a time from the dataset and generates augmented versions of the image. Each new image is independently transformed

Figure 4.3. *Examples of data augmentation in the dataset: a) original image, b) blur (cartoon) + horizontal flip + zoom in, c) reduced contrast + zoom in and d) salt & pepper noise + horizontal flip.*

from the original image by randomly selecting a transformation operation from a set of available functions. This set includes functions for adding noise, blurring, and modifying attributes such as brightness, contrast, saturation, and color temperature. To prevent extreme transformations, such as significant changes in color, the magnitude of these effects were limited by finding optimal hyperparameters. For instance, to ensure that the hazmat plates still retain their orange color, otherwise it would change the meaning of the plate. Additionally, the images were further manipulated by flipping them horizontally with 50% probability, and also by zooming in or out, and translating along x and y axis with 40% probability. Some examples can be seen in Figure 4.3. Additionally, any function that may cause warping of the images were avoided as the dimensions of hazmat plates are strictly specified and changing shape would likely cause issues.

4.5 Datasets

The data from the earlier steps were combined to create two datasets: one for training and one for testing. The training set was composed of images from Finland's road cameras, the Netherlands' traffic cameras, and creativecommons.org as well as augmented versions of these images. The test set primarily consisted of images from creativecommons.org and pictures taken with an iPhone 11, but also contained images from the same road cameras that were used for training data collection. The additional different

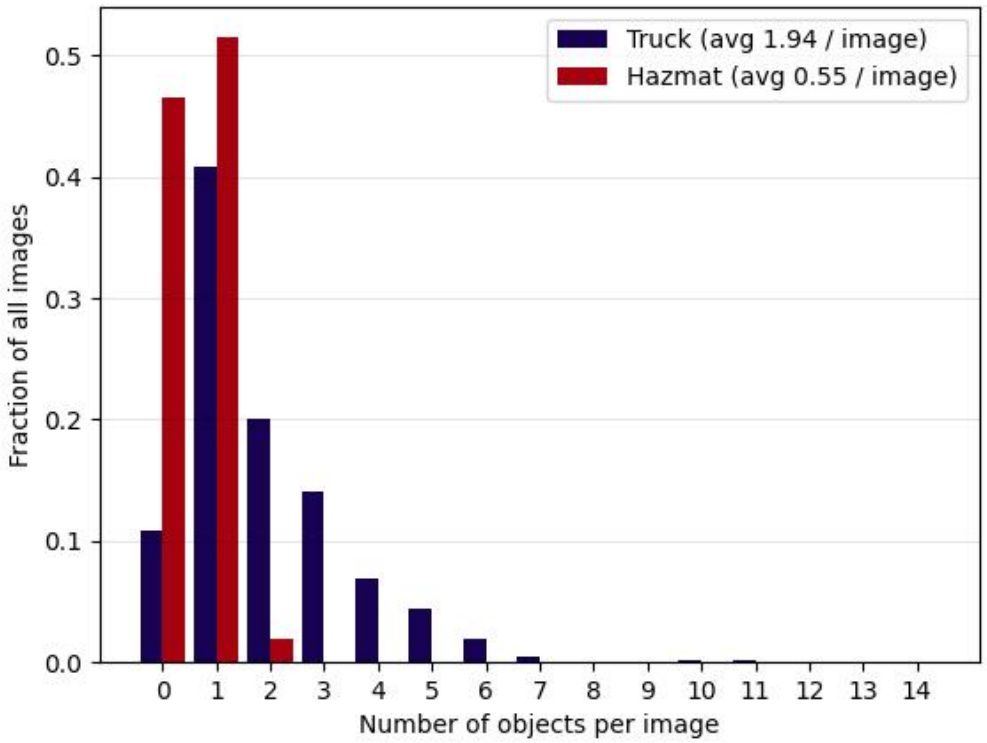
data sources were chosen for the test set in order to test the models' ability to generalize. However, it's worth noting that this test data was only used to test the object detection models and not the final application, which required data in video form. The final training set had 14 058 images (31 115 labeled instances), while the final test set had 500 images (1 505 labeled instances).

The datasets presented an issue with class imbalance, particularly in the training dataset where there were roughly 3.5 trucks for every hazmat plate as shown in Figure 4.4. This class imbalance was somewhat expected as the main roads and highways are popular routes used by trucks. On the other hand, the test dataset was not as affected by class imbalance because a significant portion of the images depicted close-up views of trucks carrying dangerous goods in their starting places or final destinations rather than busy highway traffic scenes.

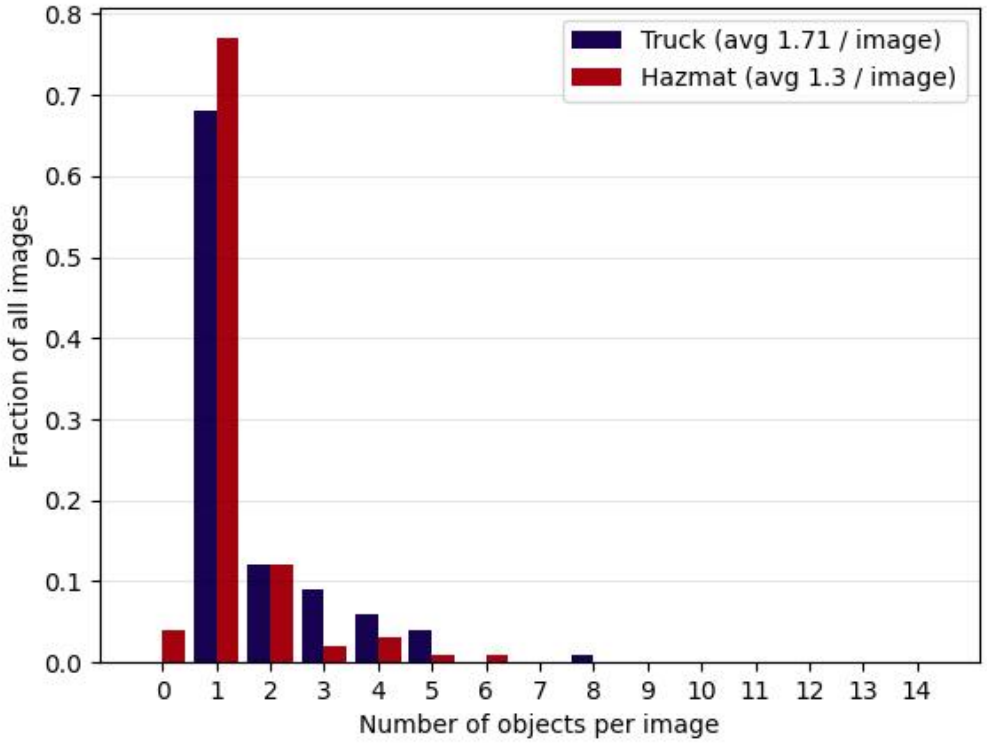
Trucks are large vehicles, but when captured by traffic cameras, they can appear quite small. On average, the bounding box for a truck in the training dataset is 6.54% of the image size. In contrast, the average size in the test dataset is 10.36% of the image size, which is nearly twice as large. This difference can be attributed to the different data sources used. The images downloaded from creativecommons.org are typically taken at a closer distance to the trucks than those taken by traffic cameras. This can be observed in Figure 4.5, where the overall distributions of the datasets are similar, but the spikes in higher values on the x-axis in the test dataset indicate the presence of close-up images.

Hazmat plates are small items, especially when compared to trucks. On average, the bounding box for a hazmat plate in the training dataset is just 0.09% of the image size. The size distribution shown in Figure 4.6 demonstrates that even the largest hazmat plates captured in close-up images of trucks are quite small. The minuscule size of hazmat plates presents some challenges for detection.

The bounding box aspect ratios in Figure 4.7 provide valuable information about the datasets. For example, the plots indicate that the majority of bounding boxes for the truck class are horizontal rectangles. This makes sense, as traffic and road cameras are usually placed alongside roads rather than directly above them, and the side profile of trucks is best captured with horizontal bounding boxes. In contrast, the bounding boxes of hazmat plates are close to being square shaped. Additionally, the plots show that there is less variation in the aspect ratios of hazmat plate bounding boxes compared to truck bounding boxes. These statistics can be useful for designing default anchors for certain object detectors.

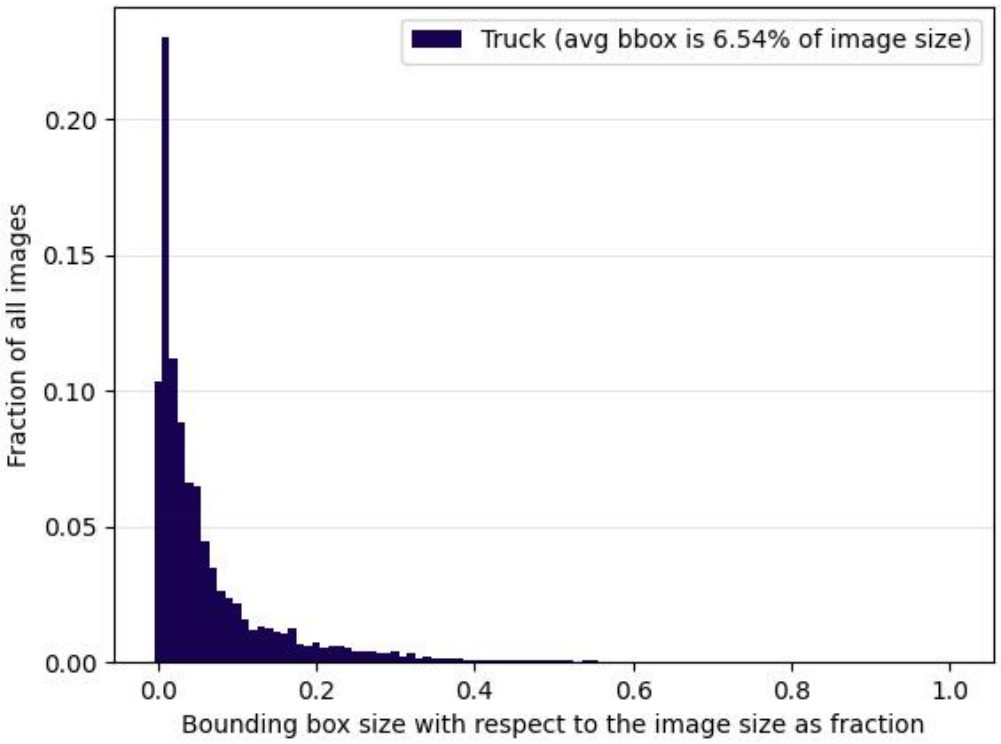


a)

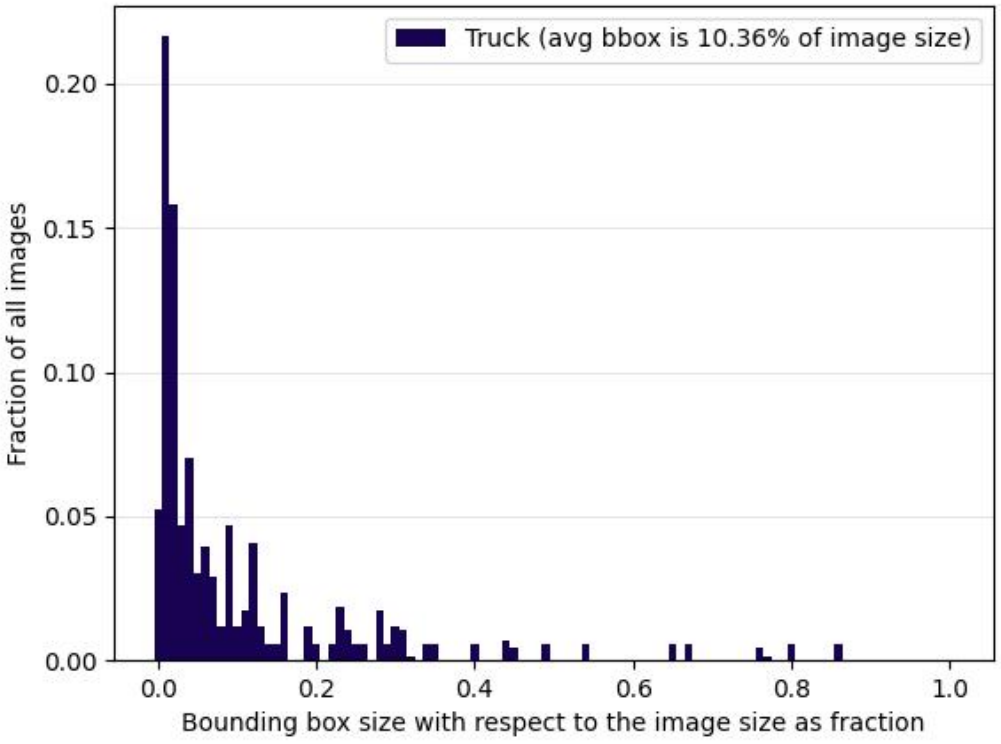


b)

Figure 4.4. *The number of objects for each class in the images in the a) training dataset and b) test dataset are presented. A significant class imbalance is particularly apparent in the training dataset.*

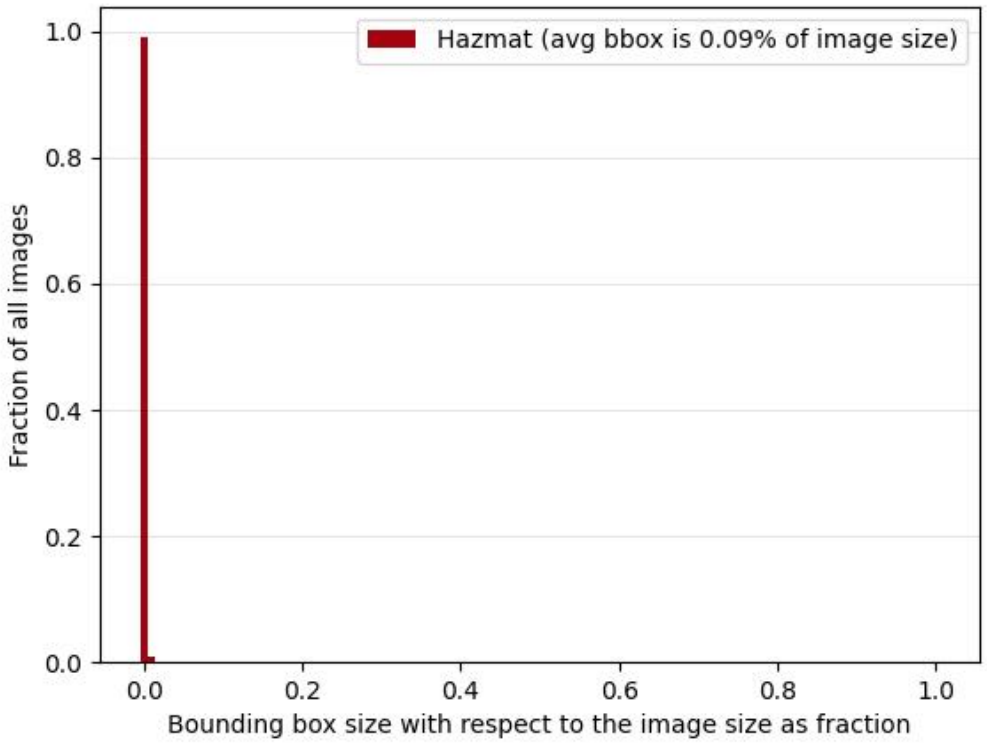


a)

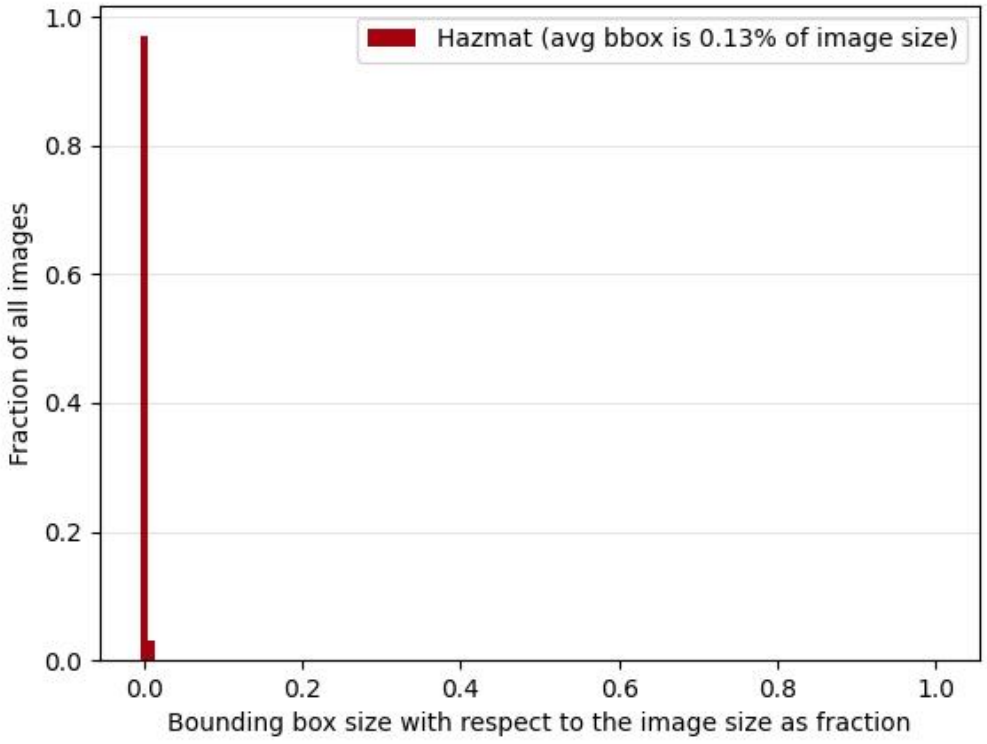


b)

Figure 4.5. *The size of the bounding boxes surrounding trucks in relation to their image sizes in the a) training data and b) test data are shown. In the training dataset, bounding boxes that take up 50% or more of the image are uncommon because the majority of the images were taken from road cameras, which are often positioned above the road to capture a broad view of traffic.*

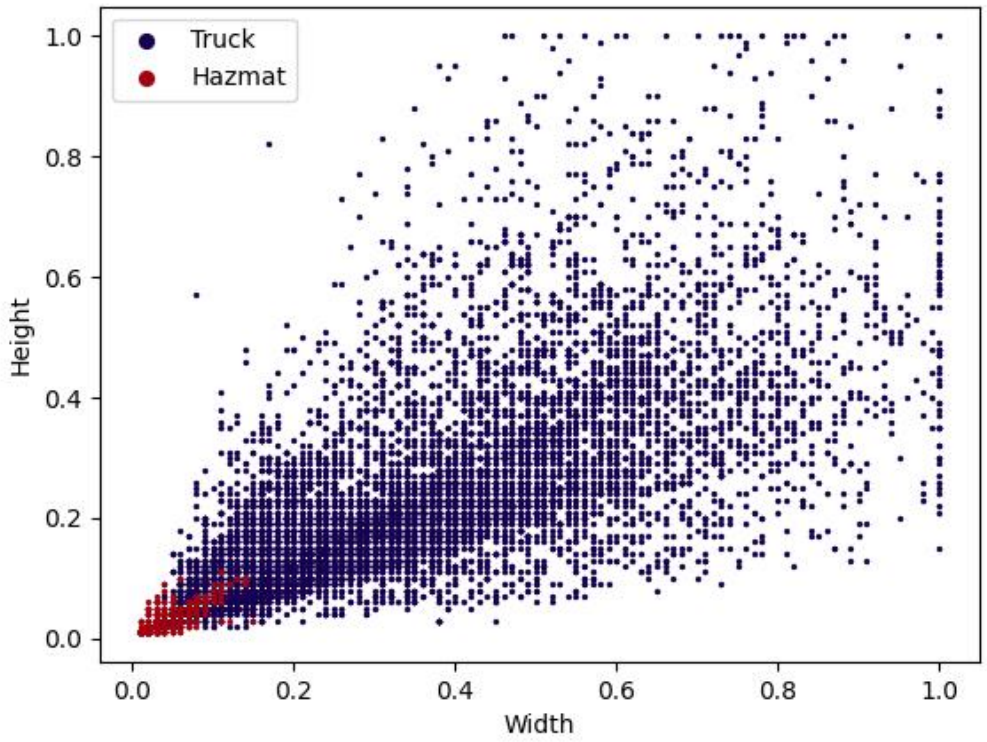


a)

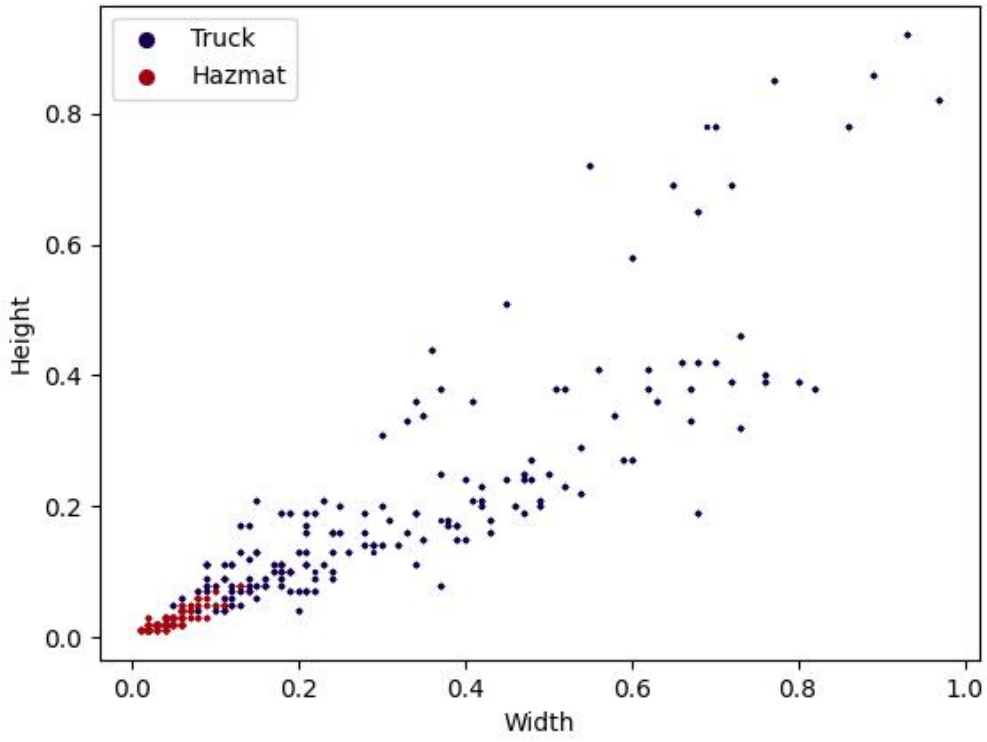


b)

Figure 4.6. *The size of bounding boxes surrounding hazmat plates in relation to the image sizes in the a) training data and b) test data are presented. The average plate size is very small, occupying only 0.09% of the image in the training data and 0.13% of the image in the test data.*



a)



b)

Figure 4.7. *The aspect ratios of bounding boxes for each class in the a) training and b) test datasets are presented. The aspect ratios of bounding boxes surrounding hazmat plates exhibit low variability due to their strictly regulated dimensions.*

There is no widely accepted definition for what constitutes a large or small dataset, as there are no specific agreed-upon thresholds. For example, in 2009, the ImageNet dataset was considered large-scale, with over 14 million images organized into more than 20 000 categories [45]. However, by 2020, the same dataset was referred to as mid-sized in a popular research paper [57]. This suggests that the size of a dataset can be relative to the current state-of-the-art technology, hardware, and algorithms. The dataset constructed in this work could be considered small, with 1562 images prior to augmentation and 14 058 images after augmentation, as it can be easily handled by consumer hardware.

5. CONSTRUCTION OF A TRUCK DATASET FOR OBJECT TRACKING

As deep learning has become more prevalent in object tracking, many modern object tracking algorithms rely on large datasets to train neural networks. However, the nature and labeling of object tracking data is different from that of object detection. As a result, the dataset collected in chapter 4 cannot be used to train object tracking algorithms. For instance, the DeepSORT algorithm, which is a popular object tracking algorithm, requires a sequence of images of the same object, ideally showcasing different angles, to learn how to associate different viewpoints with the same object [58].

VeRi [59] is a dataset for vehicle re-identification that is publicly available. It includes more than 50 000 consecutive images of 776 vehicles, but these vehicles are not limited to trucks and the majority of the data includes other types of vehicles like cars and buses. Additionally, the dataset can only be used for research purposes, making it inappropriate for this work. Consequently, a custom object tracking dataset for trucks was created. This section details the steps taken to gather the data and construct the final dataset.

5.1 Data Collection

The custom truck re-identification dataset was constructed by utilizing the same traffic cameras in the province of Gelderland in the Netherlands as described in chapter 4. While the road cameras in Finland provided useful data for the object detection dataset, they were not suitable for object tracking because they did not provide live video feed. The use of live video feed from the traffic cameras in the Netherlands was instrumental in the dataset construction process, as it allowed for the collection of image sequences in which the same truck could be captured at different points in time, as illustrated in Figure 5.1.

Figure 5.1. *A series of images of the same object taken one after the other. The sequence illustrates how the same truck appears from various perspectives. The neural network in DeepSORT must learn to associate all the different viewpoints with the same truck.*

The data collection was heavily automated using Python and included the usage of a YOLOv4 model trained on the dataset outlined in chapter 4. The Python application processed the live video frame-by-frame and utilized YOLOv4 to identify any trucks present. Any frame that included a truck was saved as an individual image and named with a timestamp to maintain the order. The application ran for hours on two separate days, which generated hundreds of image sequences and thousands of individual images.

5.2 Preprocessing and Final Dataset

The automated data collection process introduced minimal noise to the dataset, which primarily took the form of very short sequences and these were removed from the dataset manually. The primary observation from the collected images was that the vast majority of them were horizontally oriented. The original work that introduced the DeepSORT algorithm resized all images to a shape of 128x64x3, which was selected based on the nature of the data and the characteristics of the problem, which focused on pedestrian tracking [30]. This choice was logical as the bounding boxes that capture pedestrians are frequently vertical rectangles, as can be seen in Figure 5.2. However, trucks are often better captured by horizontal rectangles, making the shape of 128x64x3 impractical. As a result, the images were instead resized to 64x128x3 to better match the data.

The possibility of using data augmentation to expand the dataset was briefly considered,

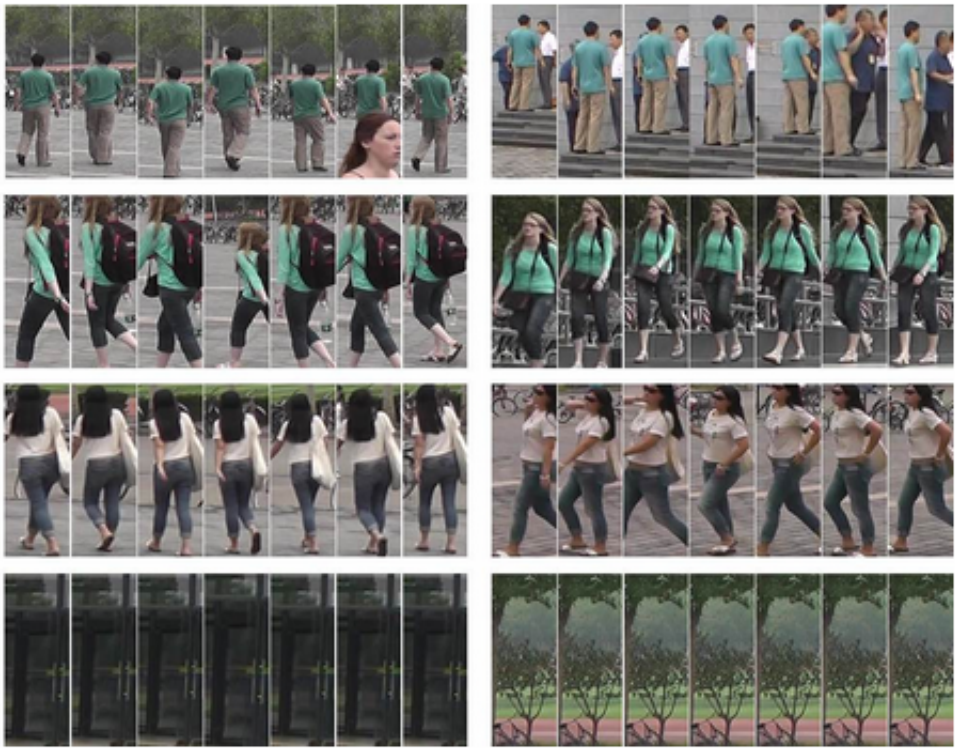


Figure 5.2. *Pedestrians naturally take up more space vertically than horizontally, making it appropriate to take images in a vertical orientation. All images in the MARS dataset have a fixed shape of 128x64x3 [60].*

but the idea was soon discarded as the augmentation methods that would be suitable for the data were very limited and challenging to implement in practice. Additionally, there was no indication from existing research that augmenting data would improve performance of object trackers similar to DeepSORT.

The final dataset, after removing noise, contains 10261 sequential images of 184 distinct trucks. On average, there are 56 images per truck. Each image has spatial dimensions of 64 rows and 128 columns in 3 color channels.

6. EXPERIMENTS

This chapter describes experiments that were carried out to find answers to the research questions and build a successful object detection application. The chapter starts by proposing a method for detecting and tracking dangerous goods on roads. This is followed by description of evaluation methods for object detection models as well as for the final application. Then, object detection models are compared and their results evaluated. Finally, object tracking selection criteria and training are briefly visited, before presenting the final evaluation results for the computer vision application.

6.1 Object Detection Model selection

The task at hand is highly practical. A successful solution could be used to monitor traffic in real-time near highways and tunnels for example. This means that the model needs to process images at a rate of around 25 FPS, preferably on an edge device such as NVIDIA's Jetson, which could be installed in the field. To meet this requirement, it is necessary to use a one-stage detector if an acceptable level of accuracy can be achieved.

Detection algorithms such as SSD and those in the YOLO family are commonly used in applications that need fast inference speeds. These algorithms are well-known and their original papers are easily accessible. They are also implemented in popular machine learning frameworks, which have extensive documentation and a large community of users for support. For these reasons, SSD and YOLO detectors were considered as the main candidates for the application.

One-stage detectors are fast but may sacrifice accuracy for speed. To determine the extent to which accuracy is compromised in this trade-off, this thesis also evaluates the performance of a state-of-the-art two-stage detector (Faster RCNN). Two-stage detectors are slower than one-stage detectors, so this evaluation helps to understand the trade-off between speed and accuracy when switching from a two-stage to a one-stage detector.

6.2 Object Detection Model Training

All of the object detection models in this thesis were trained on an Nvidia RTX 3070 GPU, using the dataset described in Chapter 4. The training was carried out using either the

Darknet or Pytorch framework, depending on the model. Each model's network weights received the same number of updates to ensure that the comparisons between the models were fair and to minimize the time required for each experiment.

6.3 Training YOLO

The YOLO object detector has undergone significant improvements over the past few years, with various versions released. In this work, YOLOv4 was used because of its performance on benchmark datasets and because it is the latest version available in the Darknet framework. The feature extractor of the detector used in this thesis was Darknet-53, which was pretrained on the ImageNet dataset.

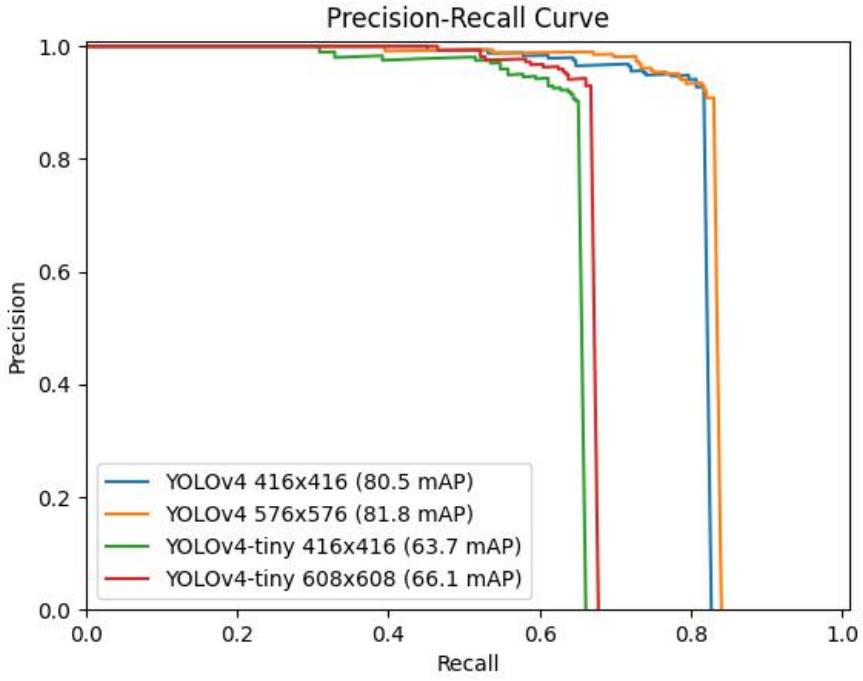


Figure 6.1. *Performance of YOLOv4 and YOLOv4-tiny models with different input sizes but otherwise the same configuration. Increasing the input image size slightly boosts the performance, but drastically reduces processing speed. YOLOv4 416x416 with OpenCV inference on RTX 3070 GPU can process images at a rate of 30 FPS, while YOLOv4 576x576 has a maximum inference speed of 15 FPS. YOLOv4-tiny models process images at whopping speeds, peaking at 73 FPS, but do not achieve the accuracy of regular YOLOv4 models.*

To compare the performance of different training and network configurations, 18 YOLOv4 and YOLOv4-tiny models were trained. To ensure that the models were fairly compared, the weights of each model were updated the same number of times (6000) during training. The training configurations were varied by adjusting learning parameters, batch size, pre-processing functions and other factors. The neural network configurations tested the

impact of the input size on performance. Larger input sizes resulted in a small improvement in performance but at the cost of reduced processing speed, which is illustrated by Figure 6.1. Custom anchors were also created for the dataset using k-means clustering of the training bounding boxes, but the default anchors for the COCO dataset performed nearly as well and did not significantly affect the training process.

6.4 Training SSD

Experiments with SSD were conducted in a similar manner to those with YOLOv4, but the Pytorch framework was used instead of Darknet. The feature extractor of the detector in this work was MobilenetV3. A total of 10 trained models were obtained from these experiments.

The SSD models did not produce good results. While the models were able to detect trucks with acceptable performance, they completely failed to identify hazmat plates. Attempts to modify the default anchors and their ratios and to try different learning parameters did not improve the ability to detect hazmat plates. The poor results may have been due to the small network input size of 320x320, but there could be other factors at play as well because a similar model, YOLOv4-tiny with an input size of 416x416, performed significantly better.

6.5 Training Faster RCNN

Faster RCNN model experiments were conducted using the Pytorch framework. A total of 13 models were trained during these experiments, which included testing custom anchors and aspect ratios, different network input sizes, and various learning parameters. The popular Resnet 50 neural network architecture was used to classify region proposals in the second stage of the detector.

The Faster RCNN models required more memory, so each model was trained with a batch size of 4, which is significantly smaller than the lowest batch size used in the YOLOv4 or SSD experiments (32). The smaller batch size resulted in the need for more training steps to achieve good performance, as the models were still underfitting at 6000 steps. Thus, all the Faster RCNN models were trained for 20 000 steps, where the learning started to plateau.

The biggest improvement in performance appeared to be due to the use of custom anchors and aspect ratios, which were generated using the k-means clustering algorithm (as shown in Figure 6.2). The anchor sizes were calculated using k = 5 to cluster the area of the bounding boxes in the training data. Similarly, the aspect ratios were calculated using k = 3 (3 aspect ratios at each scale) to cluster the aspect ratios of the bounding boxes in the training data.

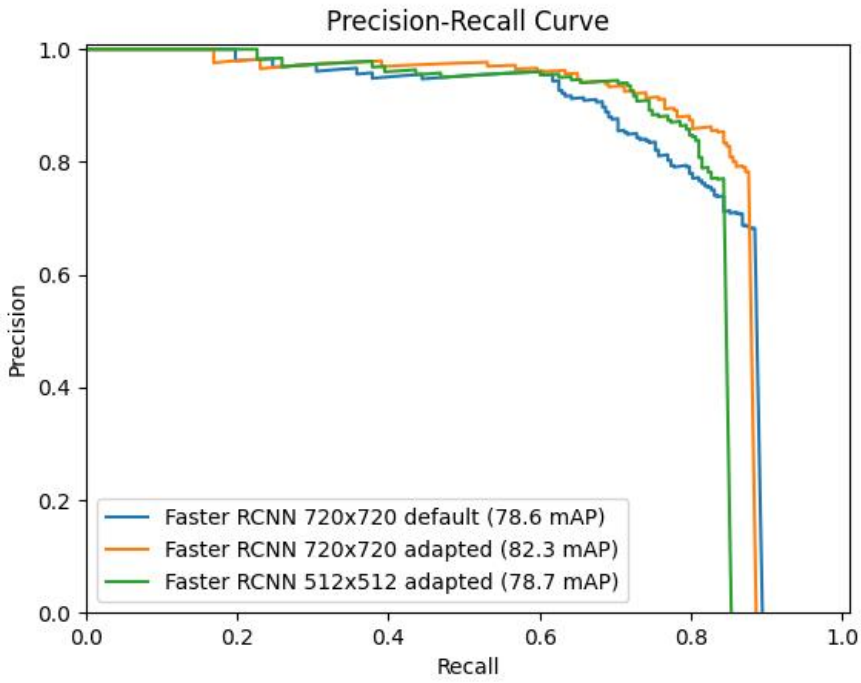


Figure 6.2. *The most significant boost in performance for faster RCNN models was achieved by changing the default anchors to data specific anchors by k-means clustering algorithm. This increased mAP value from 78.6 to 82.3. The same configuration with 512x512 network input size reached the performance of the model with default anchors, despite having smaller input size.*

Network input size mattered to a point. However, increasing the network input size after 512x512 did not increase the models' performance in a significant manner. For example, the same model configuration (anchor sizes were scaled with the increment ratio) trained with network input size of 512x512 and 720x720 produced quite identical results, the bigger model gave only a subtle boost to hazmat AP. Interestingly, models with 1:1 aspect ratio network input size outperformed models with 4:3 aspect ratio.

The network input size had some impact on performance, but increasing the size beyond 512x512 did not result in a significant improvement. For example, using the same model configuration (anchor sizes were scaled) with a network input size of 512x512 and 720x720 produced quite similar results, with only a slight increase in hazmat AP for the larger model. Interestingly, models with a 1:1 aspect ratio network input size performed better than those with a 4:3 aspect ratio.

The best performance for Faster RCNN was obtained with the network input size of 720x720, anchor sizes of [8 16 32 128 256] and aspect ratios of [0.6 2.21 0.75]. The model was optimized using SGD with momentum for 20 000 steps, resulting in an mAP of 82.3.

6.6 Object Detection Model Comparison

The experiments with the 3 object detection algorithms demonstrated the differences in their performance in various ways. The SSD models performed the worst, completely failing to detect hazmat plates (as shown in Figure 6.4). This was not entirely unexpected because it is known that SSD has difficulty with very small objects, especially when the network input size is small (320x320) [61]. Despite having a network input size almost half the size of that of Faster RCNN, YOLOv4 was the best performing model for the hazmat plate detection task. The difference between the two models was small (0.8 AP as shown in Figure 6.4), but Faster RCNN struggled with a higher number of false positives compared to YOLOv4.

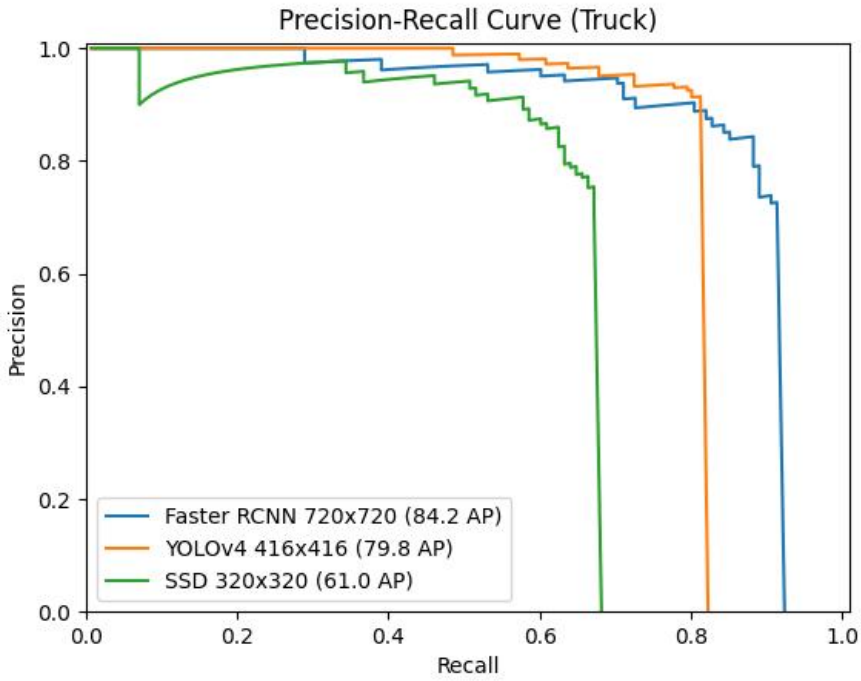


Figure 6.3. *Precision and recall values for truck detection task plotted for all the best models from each set of experiments (IoU@0.5). Faster RCNN achieved the best performance for truck detection. The model is able to detect the most of the trucks in the test data, but struggles with false positives. YOLOv4 model fails to detect trucks more often than Faster RCNN but is more robust against false positives.*

Overall, the best performing model according to mAP in the end was Faster RCNN, as Figure 6.5 and Table 6.1 suggest. The difference in overall performance between Faster RCNN and YOLOv4 was quite insignificant. The biggest difference could be found in how the two detectors performed on the truck detection task. Faster RCNN models were able to achieve a recall value of over 90%, while YOLOv4 models were stuck at 82%. Particularly detecting distant trucks gave trouble to YOLOv4 models (Figure 6.6), even though the models performed well detecting even smaller hazmat plates. As it was with

hazmat detection task, Faster RCNN was still more prone to false positives than YOLOv4.

Overall, the model with the highest mAP was Faster RCNN, as shown in Figure 6.5. The difference in overall performance between Faster RCNN and YOLOv4 was quite insignificant. The biggest difference was seen in how the two detectors performed on the truck detection task (as shown in Figure 6.3), with Faster RCNN achieving a recall value of over 90% and YOLOv4 stuck at 82%. YOLOv4 models had difficulty detecting distant trucks, even though they performed well at detecting small hazmat plates. Figure 6.6 shows how Faster RCNN succeeds to detect trucks of different sizes while YOLOv4 and SSD only detect the largest. Like in the hazmat detection task, Faster RCNN also was more prone to false positives in the truck detection task than YOLOv4.

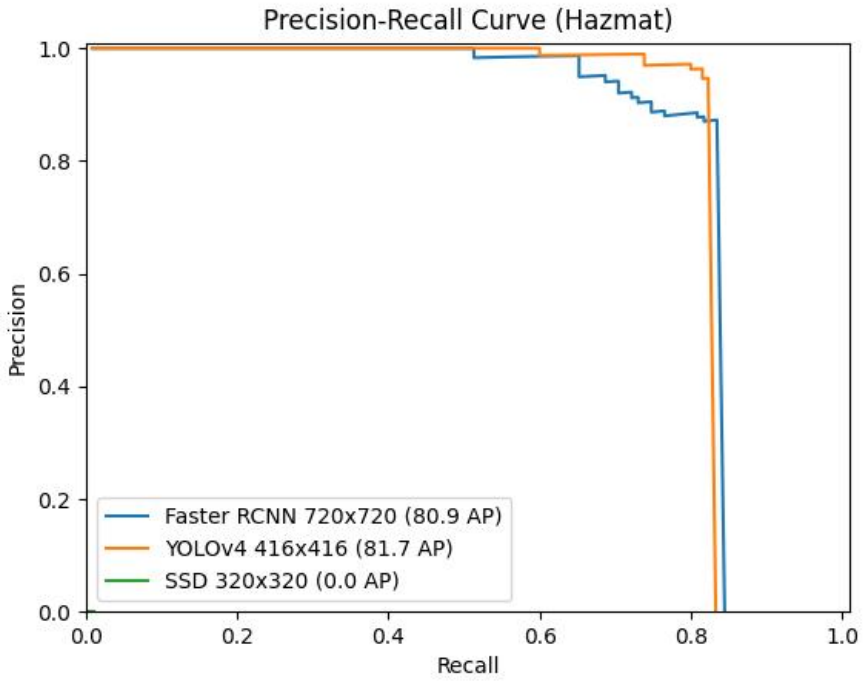


Figure 6.4. *Precision and recall values for hazmat plate detection task plotted for all the best models from each set of experiments (IoU@0.5). YOLOv4 achieves the best AP for detecting hazmat plates, outperforming the state-of-the-art two-stage detector Faster RCNN. Like with the truck detection task, Faster RCNN had more false positives when detecting hazmat plates. SSD was not able to learn to detect any hazmat plates.*

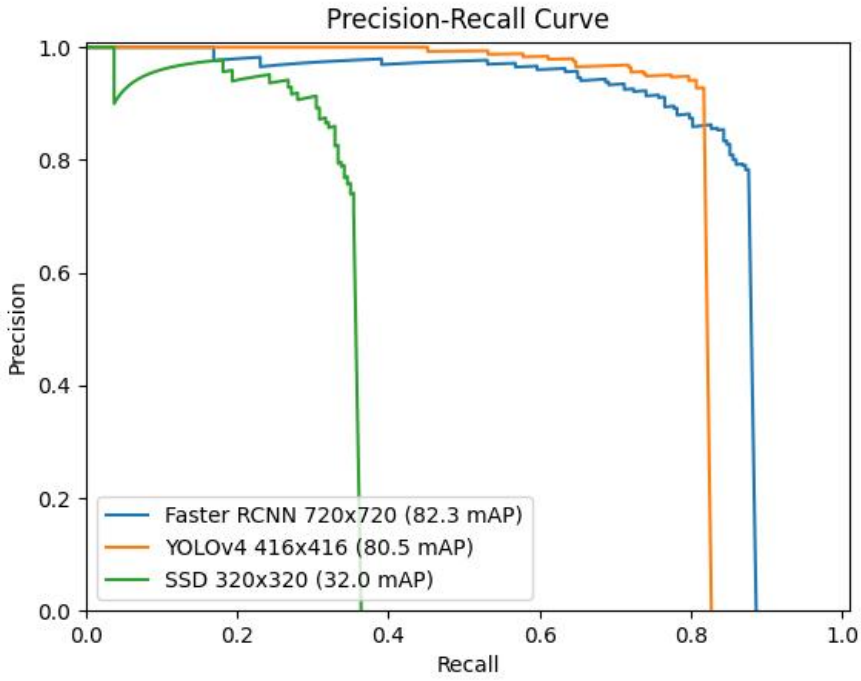


Figure 6.5. *Precision and recall values plotted for all the best models from each set of experiments (IoU@0.5). Overall, Faster RCNN and YOLOv4 perform quite similarly based on the mAP metric. However, YOLOv4 may be the preferred choice due to its lower number of false positives.*

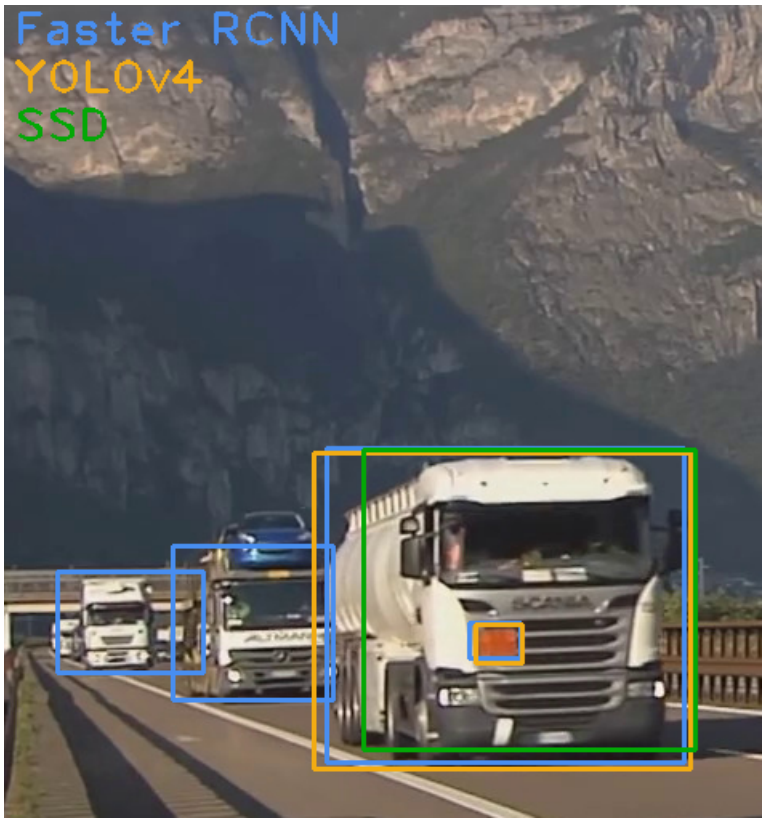


Figure 6.6. *YOLOv4 models are able to detect small hazmat plates well, but sometimes have difficulty with small to medium trucks. In the figure, YOLOv4 is able to detect the first truck in the front, but fails to detect the others, while Faster RCNN detects all the trucks. However, the predicted bounding boxes by YOLOv4 generally fit better around the objects of interest. For the predictions in the image, an IoU threshold of 0.5 was used.*

Detector	AP_{T_M}	AP_{T_L}	AP_{H_S}	AP_{H_M}	AP_T	AP_H	mAP	Inference (ms)
YOLOv4 416x416	8.4	76.0	56.1	**84.9**	79.8	**81.7**	80.5	33
YOLOv4 576x576	6.3	75.7	53.0	83.6	83.7	79.3	81.8	67
YOLOv4-tiny 416x416	3.1	65.3	45.5	54.6	67.1	59.3	63.7	**14**
YOLOv4-tiny 608x608	5.0	59.8	54.4	62.8	64.4	68.1	66.1	24
F-RCNN 512x512	12.2	76.9	57.4	67.1	81.4	76.7	78.7	48
F-RCNN 720x720	**36.4**	**77.2**	**63.4**	69.5	**84.2**	80.9	**82.3**	50
SSD 320x320	0.0	63.3	0.0	0.0	61.0	0.0	32.0	20

Table 6.1. *Summary of the detection results. Average precision (AP) results were calculated with IoU = 0.5. The first subscript denotes class truck (T) or hazmat (H). The second subscript indicates area of objects, where small (S) is smaller than 32^2 pixels, medium (M) is bigger than 32^2 but smaller than 96^2 pixels, and large (L) is bigger than 96^2 pixels. Inference speed measures a time for a detector to process one frame. The inference speed value is calculated as an average over thousands of frames.*

However, mean average precision is just one way to compare the models. Another important performance metric for this thesis was the inference speed of the models, which was measured on an RTX 3070 GPU. The results in Table 6.1 show that YOLOv4-tiny with a network input size of 416x416 had the best inference speed at 14 milliseconds per frame, or 73 FPS. The other one-stage detector, SSD, had an inference speed of 20 milliseconds per frame, or 50 FPS. As expected, smaller neural network architectures had faster inference speed but lower accuracy. YOLOv4 with a network input size of 416x416 had a real-time inference speed of 33 milliseconds per frame, or 30 FPS, while being competitive with Faster RCNN in terms of accuracy. The fastest Faster RCNN model had an inference speed of 48 milliseconds per frame, or 21 FPS. It is worth noting that these results should be taken as a rough comparison of inference speed, as YOLOv4 used inference functions from the OpenCV framework while SSD and Faster RCNN used functions from the Pytorch framework, which may have caused small differences in performance.

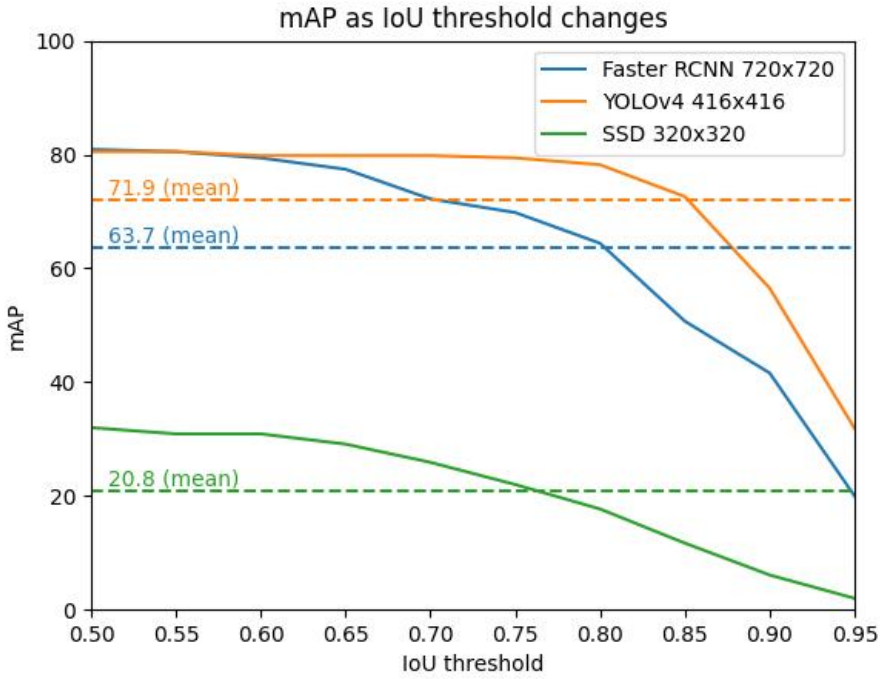


Figure 6.7. *Illustration of how mAP changes when IoU threshold to be considered a true positive is changed. This shows the localization accuracy for the predicted bounding boxes by detectors. Performance of YOLOv4 remains quite stable all the way to threshold of 0.8, while the performance of Faster RCNN and SSD start to deteriorate early, implying that the bounding boxes predicted by YOLOv4 are by average closest to the ground truths.*

So far, the statistics discussed have focused on the ability of the detectors to accurately locate objects of interest in images and the speed of neural computations, which are the most important factors to consider when choosing a detector for the specific application. However, it is also important to consider the ability of the detectors to adjust their predicted bounding boxes to match the ground truths. If the predicted bounding boxes do not fit well,

there is a risk that a hazmat plate detection will be outside of a truck detection and will not be considered a dangerous truck. Well-fitted bounding boxes can also increase people's confidence in the computer vision system when visualized. As shown in Figure 6.7, as the IoU threshold increases, the mAP decreases. YOLOv4 had the most accurate bounding boxes, with performance remaining stable up to an IoU threshold of 0.8, after which it began to decline more rapidly. In contrast, Faster RCNN saw a decline in performance after an IoU threshold of 0.6. This can be seen in Figure 6.6, where the smallest bounding box predicted by Faster RCNN is not well-fitted around the truck and would not be counted as a prediction with higher IoU threshold values.

6.7 Lessons Learned

Training an object detector and plotting a precision-recall curve can give an idea of how well the detector performs, but does not provide much insight into the inner workings of the detector or explain why its performance is at a certain level. To gain a deeper understanding of the detector, it is helpful to visualize detections and network weights. Visualizing detections allows one to see what the detector is seeing, while inspecting neural network weights can reveal problems that occurred during training that may not be apparent otherwise.

Figure 6.8. *Visual inspection of the model's detections can highlight the types of false positives and negatives that the model has difficulty with, which can guide additional data collection. In this example, YOLOv4 416x416 produced a false positive detection of a truck with high confidence, in addition to correct detections..*

Figure 6.9. *The Faster RCNN model incorrectly identified an advertisement on the side of a truck as a hazmat plate. Objects that are shaped like orange rectangles, similar to hazmat plates, can easily lead to false detections.*

After training the models, their detections on the test data were visualized. A common theme in the visualizations was that certain objects often resulted in false detections. The most common false truck detections included close-up road signs (Figure 6.8) and random cylinder-shaped objects. The most common objects falsely detected as hazmat plates were mostly different types of orange square-shaped paintings on trucks (Figure 6.9), as well as vehicles carrying box-shaped items such as crates that are similar in color.

6.8 Impact of Labeling

The dataset in 4 has two classes: truck and hazmat, which are necessary for the proposed method. Therefore, an association step was needed to identify trucks transporting dangerous goods by using the hazmat plate and truck detections as sources of information. The same data was also labeled so that there is only one class: dangerous truck. This allows for end-to-end detection of dangerous goods.

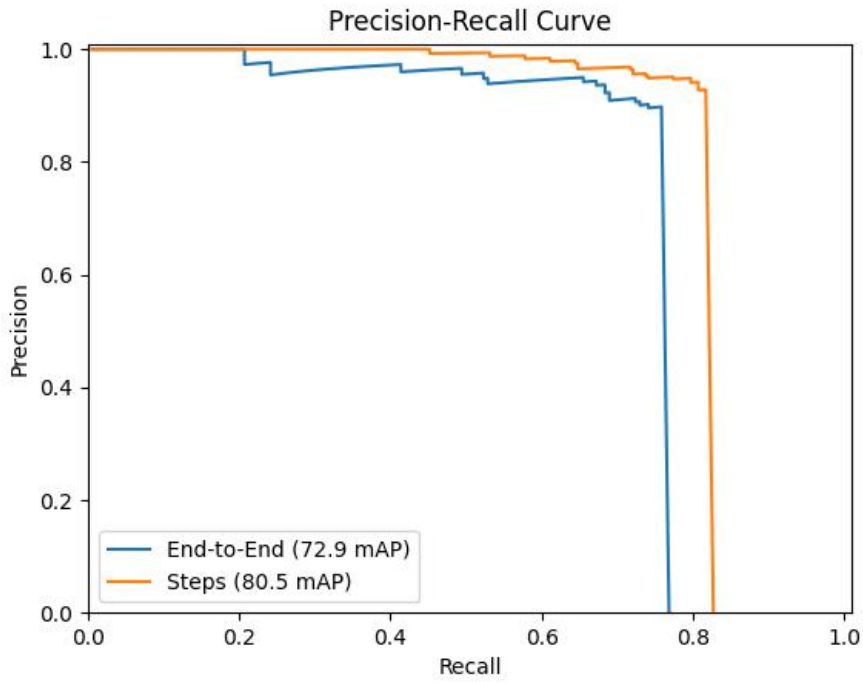


Figure 6.10. *The performance difference of YOLOv4 416x416 trained end-to-end to detect dangerous goods versus the same model using the proposed method. The proposed method outperforms the end-to-end detector by a big margin of 7.6 mAP.*

The training results showed a huge difference between the mAP statistics between the models trained on the same data but labeled differently. The models trained end-to-end (1 class) performed way worse than the models trained to detect trucks and hazmat plates. The difference in performance was significant 7.6 mAP (Figure 6.10). A plausible explanation for the worse performance of the end-to-end trained models is that the amount of data is insufficient. The visual difference between a truck and a dangerous truck is essentially just the orange hazmat plate. Trucks may carry other kind of plates as well, or the painting of the truck may sometimes have shapes and colors resembling the hazmat plate. Learning these features with the end-to-end training method may take longer and require more data.

The results of the training showed a significant difference in the mAP statistics between the models that were trained on the same data but labeled differently. The models trained end-to-end with 1 class performed much worse than the models trained to detect trucks and hazmat plates and follow the proposed method. The difference in performance was a significant 7.6 mAP points, as Figure Figure 6.10 shows. One possible reason for the poorer performance of the end-to-end trained models is that there was not enough data to learn to discriminate between a regular heavy duty truck and a truck with hazmat plates attached. The only discriminating factor is the presence of orange hazmat plates. However, trucks may also have other types of plates, or the paint on the truck may sometimes

resemble a hazmat plate in shape and color. Learning these features with the end-to-end training method may take longer and require more data.

6.9 Object Tracker Selection

When selecting an object tracker, it was important to consider real-time processing speed on a consumer device, as it was for selecting an object detection model. SORT (Simple Online and Realtime Tracking), a popular object tracker, met this requirement and was chosen as one of the candidates. One disadvantage of SORT is that it can struggle with identity switches when multiple objects are occluded by each other. However, the problem this thesis aims to solve rarely involves tracking more than one object at a time, and the object is rarely occluded for extended periods of time. Therefore, SORT was considered a good choice for this case.

To determine if SORT was suitable for the problem at hand, an upgraded version called DeepSORT was chosen as an additional candidate. DeepSORT works similarly to SORT but includes a neural network to assist with object identification and prevent identity switches. The added neural network adds processing time, but the tracker still runs fast enough to meet the real-time processing requirement.

6.10 DeepSORT Training

As mentioned in section 6.9, DeepSORT uses a neural network to re-identify objects. This means that the weights of the neural network must be re-trained for a new dataset. The neural network in the original DeepSORT paper [30] was trained on the MARS and Market 1501 datasets, which are focused on person re-identification.

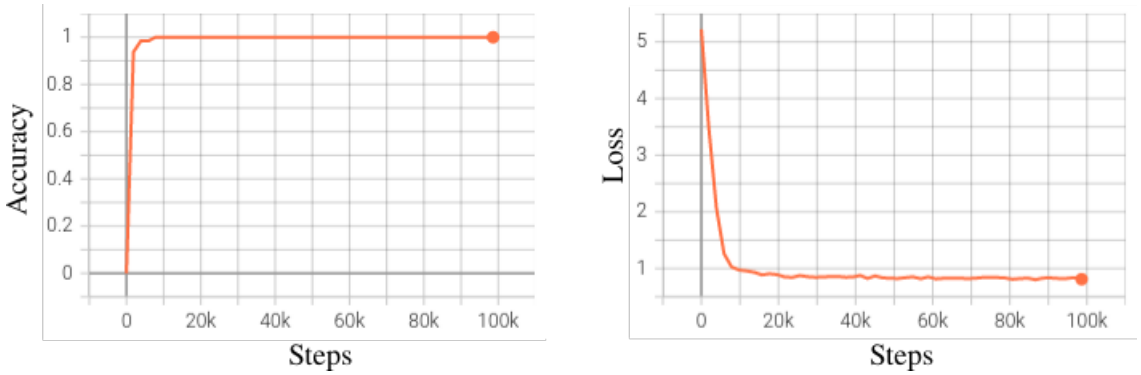


Figure 6.11. *Evolution of the DeepSORT's neural network's validation accuracy and loss during the training. The learning plateaus around 20 000 steps.*

New weights for truck re-identification were obtained by training the neural network on the data described in Chapter 5. The network weights were updated 100,000 times with a batch size of 64 during training, resulting in approximately 1,562 epochs. The training was

conducted within the Tensorflow framework, using code provided by the original authors of the DeepSORT paper as a base. The network weights were optimized with the Adam optimization algorithm, and the loss was calculated using cross-entropy. As shown in Figure 6.11, the validation accuracy and loss became stable after only 20,000 steps. Weights trained for 100,000 steps achieved a similar performance to the weights trained for trained for 20 000 steps.

6.11 Results

The final evaluations were conducted using 16 videos of varying length, the shortest being 20 seconds and the longest being 50 seconds long. Each video contained footage of highway traffic under various weather and lighting conditions, shot from different camera angles and featuring a wide range of vehicles, including trucks transporting dangerous goods. These videos were recorded manually to avoid bias that would have been introduced by automatic recording using the object detectors. This naturally limited the amount of data that could be collected in a reasonable time frame. Another limiting factor was the data labeling process, which also relied heavily on manual labor.

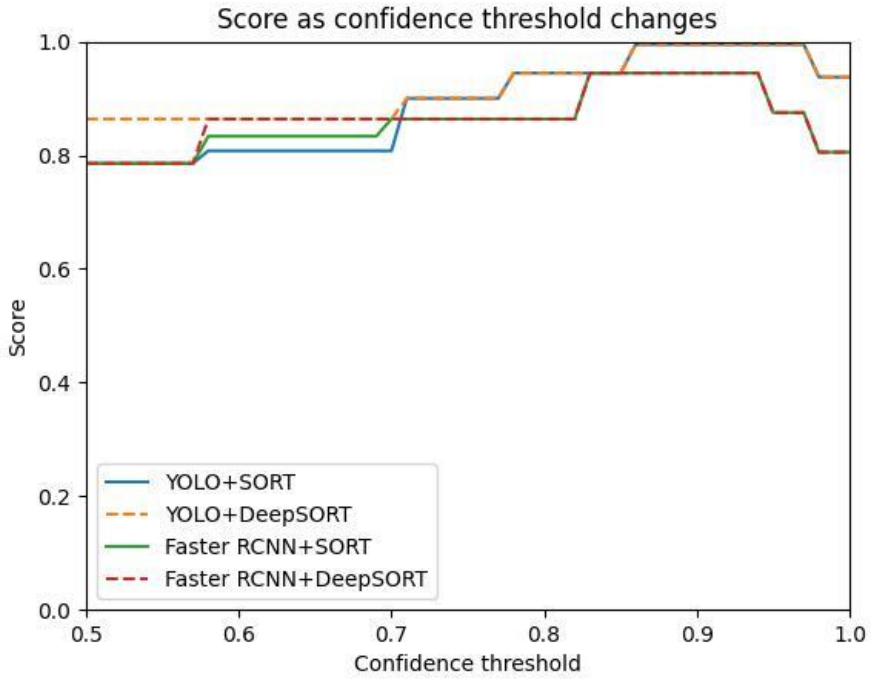


Figure 6.12. *Final scores for the evaluated applications (IoU @0.5). YOLOv4 achieves the perfect score with confidence threshold being bigger than 0.86 and smaller than 0.98. Faster RCNN fails to achieve the perfect score due to lower thresholds producing false positives and larger thresholds resulting in false negatives.*

The results of the evaluations (Figure 6.12) showed very promising statistics for appli-

cations using YOLOv4 as the object detector. The highest possible score was achieved when the detector was accompanied by either DeepSORT or SORT. However, Faster RCNN as a detector did not achieve the highest possible score. At lower confidence thresholds, Faster RCNN was successful in detecting all of the ground truth objects, but these detections were accompanied by some false positives. Confidence thresholds higher than 0.94 eliminated the false positives, but this higher confidence level also resulted in some missed detections.

The results suggest that the object detector plays a more significant role than the object tracker for this specific use case. This is evident as the score graph follows a similar path most of the time regardless of the object tracker used. This makes sense in practice, as it is unlikely to encounter a crowded scene of trucks transporting dangerous goods on a highway, making it easy for non-deep learning-based object trackers to track objects. A performance difference can be observed at lower thresholds, where DeepSORT outperforms SORT. In this case, DeepSORT is more robust against sudden movements of the bounding boxes, which are more likely to occur when the confidence is low.

Combining an object detector with SORT did not significantly increase the processing time compared to using the object detector alone. The application still achieved real-time processing speeds on an RTX 3070 GPU with YOLOv4 and OpenCV inference. On average, the final application with YOLOv4 and SORT took about 34.5 ms to process a frame, only about 1.5 ms more than it did with just the object detector (29 FPS vs 30 FPS). However, if the tracker needed to constantly track a large number of objects in crowded scenes, the increase in processing time would likely be greater. This also applies to DeepSORT, which achieved real-time processing speeds with YOLOv4 in the same setup. However, the application with YOLOv4 and DeepSORT saw a larger increase in processing time compared to using SORT, as expected. On average, this application took about 40 ms to process a single frame, an increase of 7 ms from the 33 ms achieved by the object detector alone (25 FPS vs 30 FPS). In addition to the number of objects being tracked, the number of appearance descriptors kept per detection greatly impacts the time it takes to complete the calculations. An unlimited number of such descriptors can improve tracking performance, but it comes at the cost of decreased speed. In practice, using an unlimited number of descriptors slowed the application down to single digits (below 10 FPS) from the theoretical 30 FPS, with no significant benefits. For the evaluation with DeepSORT, 100 appearance descriptors were kept per detection, as in the original DeepSORT paper [30].

It is evident that the processing time increases when object detectors are paired with object trackers, which is expected. However, the overall processing speed can be improved with little to no penalty by skipping the detection part every few frames and letting the tracker do the work. An additional test was conducted in this thesis where the detection part was skipped every 3rd frame, which allowed the application with Faster RCNN and

DeepSORT to run in real-time without affecting the final score. However, this does not necessarily apply in all cases and may make the application less robust.

Finally, the applications with YOLOv4 as the object detector were tested on a NVIDIA Jetson Xavier NX module to determine if real-time analysis is possible with these applications on modern edge devices. Using the OpenCV framework for inference resulted in slow processing speeds, averaging 5 FPS or 200 ms per frame. To improve the speed, the YOLOv4 darknet model and its weights were converted into TensorRT format, which enables optimized deep learning inference on NVIDIA's CUDA platforms. When the optimized model was compiled on the edge device, there was a significant speed increase, with YOLOv4 and SORT averaging 26 FPS or approximately 38.5 ms per frame, demonstrating that the application can run in real-time on modern edge devices.

7. CONCLUSION

This thesis explored the feasibility of using deep learning to reliably detect and track trucks transporting dangerous goods on roads in efficient manner. The study began by constructing a dataset, where the main focus was on collecting images of trucks with orange hazmat plates attached to the front and to the back. Then, a simple method for detecting dangerous goods was proposed, involving the following steps: 1) detecting trucks and hazmat plates, 2) associating the detected plates with trucks based on their 2D coordinates, and 3) tracking trucks associated with hazmat plates. Evaluation of the method on test videos showed that deep learning can be effective for this task, with the final application achieving the maximum score on the test videos. The results also suggest that the proposed solution can be efficient, reaching real-time processing speeds on a modern edge-device.

This thesis presents a method for detecting and tracking trucks transporting hazardous materials using deep learning methods that can run efficiently on modern edge devices. This can be used to quickly alert authorities of potential accidents involving dangerous goods in areas with high population density or in confined spaces such as tunnels. By integrating this method with additional software logic and deploying it with existing road cameras or new hardware near tunnel entrances, it has the potential to save lives in the event of accidents involving hazardous materials.

The scope of this thesis was restricted to detecting and tracking hazardous cargo. Identifying the specific hazardous substance based on the numbers on hazmat plates was not addressed in this work. However, in the event of an accident involving hazardous materials, knowledge of the substance being carried can be crucial for first responders to better prepare for assistance. Therefore, researching the identification of carried substances should be a priority in future studies.

REFERENCES

[1] Dandie, B. Dangerous Goods in Tunnels: Literature Review. eng. (2019).

[2] Publishing, O. *Safety in Tunnels: Transport of Dangerous Goods through Road Tunnels*. eng. Paris: OECD Publishing, 2001. ISBN: 926419651X.

[3] Caliendo, C. and De Guglielmo, M. L. Quantitative Risk Analysis on the Transport of Dangerous Goods Through a Bi-Directional Road Tunnel. eng. *Risk analysis* 37.1 (2017), pp. 116–129. ISSN: 0272-4332.

[4] Zhao, L., Qian, Y., Hu, Q.-M., Jiang, R., Li, M. and Wang, X. An analysis of hazardous chemical accidents in China between 2006 and 2017. eng. *Sustainability (Basel, Switzerland)* 10.8 (2018), pp. 2935–. ISSN: 2071-1050.

[5] Vuilleumier, F., Weatherill, A. and Crausaz, B. Safety aspects of railway and road tunnel: Example of the Lötschberg railway tunnel and Mont-Blanc road tunnel. eng. 17.2 (2002), pp. 153–158. ISSN: 0886-7798.

[6] Bęczkowska, S., Choromański, W. and Grabarek, I. Risk and human factor in carriage of dangerous goods by road. eng. *Problemy Transportu* 15.4 (2020), pp. 19–28. ISSN: 1896-0596.

[7] Kallio, R. *Vaarallisten aineiden kuljetukset ja teiden kunnossapito – toimintatapa onnettomuustilanteissa*. fin. Teollisuustalouden laitos - Department of Industrial Management, 2012.

[8] Lee, K. B. and Shin, H. S. An Application of a Deep Learning Algorithm for Automatic Detection of Unexpected Accidents Under Bad CCTV Monitoring Conditions in Tunnels. eng. *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML)*. Piscataway: IEEE, 2019, pp. 7–11. ISBN: 1728129141.

[9] Roth, P. M., Kostinger, M., Wohlhart, P., Bischof, H. and Birchbauer, J. A. Automatic Detection and Reading of Dangerous Goods Plates. eng. *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*. IEEE, 2010, pp. 580–585. ISBN: 9781424483105.

[10] McBob, C. *Tesla FSD Beta Update 10.69.25.2 First Drive and Initial Impressions | 2022.44.30.10*. 2023. URL: https://www.youtube.com/watch?v=tY7rPjQw6u0 (visited on 01/22/2023).

[11] Adhikari, B. and Huttunen, H. Iterative Bounding Box Annotation for Object Detection. eng. Tampere University. IEEE, 2021.

[12] *Hasty https://hasty.ai/*. https://hasty.ai/. Accessed: 2022-11-20.

[13] Chien-Yao, W., Bochkovskiy, A. and Hong-Yuan, M. L. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. eng. *arXiv.org* (2022). ISSN: 2331-8422.

[14] Girshick, R., Donahue, J., Darrell, T. and Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. eng. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014, pp. 580–587. ISBN: 9781479951185.

[15] Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. eng. *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916. ISSN: 0162-8828.

[16] Fast R-CNN. eng. Vol. 2015. IEEE International Conference on Computer Vision. IEEE. IEEE, 2015, pp. 1440–1448.

[17] Ren, S., He, K., Girshick, R. and Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems*. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama and R. Garnett. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[18] He, K., Gkioxari, G., Dollar, P. and Girshick, R. Mask R-CNN. eng. *IEEE transactions on pattern analysis and machine intelligence* 42.2 (2020), pp. 386–397. ISSN: 0162-8828.

[19] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338.

[20] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C.-Y. and Berg, A. C. SSD: Single Shot MultiBox Detector. *CoRR* abs/1512.02325 (2015). arXiv: 1512.02325. URL: http://arxiv.org/abs/1512.02325.

[21] Redmon, J., Kumar Divvala, S., Girshick, R. B. and Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[22] Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q. and Tian, Q. CenterNet: Keypoint Triplets for Object Detection. eng. *2019 IEEE/CVF INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV 2019)*. Vol. 2019-. IEEE International Conference on Computer Vision. IEEE. NEW YORK: IEEE, 2019, pp. 6568–6577. ISBN: 9781728148038.

[23] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. eng. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 779–788. ISBN: 9781467388511.

[24] He, K., Zhang, X., Ren, S. and Sun, J. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

[25] Simonyan, K. and Zisserman, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. DOI: 10.48550/ARXIV.1409.1556. URL: https://arxiv.org/abs/1409.1556.

[26] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Hartwig, A. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. eng. *arXiv.org* (2017). ISSN: 2331-8422.

[27] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L. and Dollár, P. *Microsoft COCO: Common Objects in Context*. 2014. DOI: 10.48550/ARXIV.1405.0312. URL: https://arxiv.org/abs/1405.0312.

[28] Xu, Y., Zhou, X., Chen, S. and Li, F. Deep learning for multiple object tracking: a survey. eng. *IET computer vision* 13.4 (2019), pp. 355–368. ISSN: 1751-9632.

[29] Ciaparrone, G., Luque Sánchez, F., Tabik, S., Troiano, L., Tagliaferri, R. and Herrera, F. Deep learning in video multi-object tracking: A survey. eng. *Neurocomputing (Amsterdam)* 381 (2020), pp. 61–88. ISSN: 0925-2312.

[30] Wojke, N., Bewley, A. and Paulus, D. Simple Online and Realtime Tracking with a Deep Association Metric. (2017). DOI: 10.48550/ARXIV.1703.07402. URL: https://arxiv.org/abs/1703.07402.

[31] Wojke, N., Bewley, A. and Paulus, D. Simple online and realtime tracking with a deep association metric. eng. *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649. ISBN: 9781509021758.

[32] Koch, G., Zemel, R. and Salakhutdinov, R. Siamese Neural Networks for One-shot Image Recognition. 2015.

[33] Ma, C., Yang, C., Yang, F., Zhuang, Y., Zhang, Z., Jia, H. and Xie, X. Trajectory Factory: Tracklet Cleaving and Re-Connection by Deep Siamese Bi-GRU for Multiple Object Tracking. eng. *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2018, pp. 1–6. ISBN: 9781538617373.

[34] Lee, S. and Kim, E. Multiple Object Tracking via Feature Pyramid Siamese Networks. eng. *IEEE access* 7 (2019), pp. 8181–8194. ISSN: 2169-3536.

[35] Milan, A., Rezatofighi, S. H., Dick, A., Reid, I. and Schindler, K. Online Multi-Target Tracking Using Recurrent Neural Networks. eng. *Proceedings of the ... AAAI Conference on Artificial Intelligence*. Vol. 31. 1. 2017.

[36] Sadeghian, A., Alahi, A. and Savarese, S. Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies. eng. *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 300–311. ISBN: 9781538610329.

[37] Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems. eng. *Journal of basic engineering* 82.1 (1960), pp. 35–45. ISSN: 0098-2202.

[38] Kuhn, H. W. The Hungarian method for the assignment problem. eng. *Naval research logistics* 52.1 (2005), pp. 7–21. ISSN: 0894-069X.

[39] Leal-Taixé, L., Anton, M., Reid, I., Roth, S. and Schindler, K. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. eng. *arXiv.org* (2015). ISSN: 2331-8422.

[40] Geiger, A., Lenz, P., Stiller, C. and Urtasun, R. Vision meets robotics: The KITTI dataset. eng. *The International journal of robotics research* 32.11 (2013), pp. 1231–1237. ISSN: 0278-3649.

[41] Bernardin, K. and Stiefelhagen, R. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. eng. *EURASIP journal on image and video processing* 2008.1 (2008), pp. 1–10. ISSN: 1687-5176.

[42] Performance Measures and a Data Set for Multi-target, Multi-camera Tracking. eng. *Computer Vision – ECCV 2016 Workshops*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 17–35. ISBN: 9783319488806.

[43] Austerlitz, H. *Data acquisition techniques using PCs*. eng. San Diego, California: Academic Press, 2003. ISBN: 9786611046255.

[44] Findley, D. J., Cunningham, C. M. and Hummer, J. E. Comparison of mobile and manual data collection for roadway components. eng. (2010).

[45] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[46] Yoshihashi, R., Kawakami, R., Iida, M. and Naemura, T. Bird detection and species classification with time-lapse images around a wind farm: Dataset construction and evaluation. eng. *Wind energy (Chichester, England)* 20.12 (2017), pp. 1983–1995. ISSN: 1095-4244.

[47] Varkarakis, V. and Corcoran, P. Dataset Cleaning - A Cross Validation Methodology for Large Facial Datasets using Face Recognition. eng. *2020 12th International Conference on Quality of Multimedia Experience, QoMEX 2020*. International Workshop on Quality of Multimedia Experience. IEEE. NEW YORK: IEEE, 2020, pp. 1–6. ISBN: 1728159652.

[48] Wang, F., Chen, L., Li, C., Huang, S., Chen, Y., Qian, C. and Loy, C. C. The Devil of Face Recognition Is in the Noise. eng. *COMPUTER VISION - ECCV 2018, PT IX*. Vol. 11213. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 780–795. ISBN: 3030012395.

[49] Nanni, L., Paci, M., Brahnam, S. and Lumini, A. Comparison of Different Image Data Augmentation Approaches. eng. (2021-11-27).

[50] Shorten, C. and Khoshgoftaar, T. M. A survey on Image Data Augmentation for Deep Learning. eng. *Journal of big data* 6.1 (2019), pp. 1–48. ISSN: 2196-1115.

[51] Nanni, L., Paci, M., Brahnam, S. and Lumini, A. Feature transforms for image data augmentation. eng. *Neural computing & applications* 34.24 (2022), pp. 22345–22356. ISSN: 0941-0643.

[52] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. *Going Deeper with Convolutions*. 2014. DOI: 10.48550/ARXIV.1409.4842. URL: https://arxiv.org/abs/1409.4842.

[53] Su, H., Deng, J. and Fei-Fei, L. Crowdsourcing annotations for visual object detection. (Jan. 2012), pp. 40–46.

[54] Tesla. *Tesla AI Day 2022*. 2022. URL: https://www.youtube.com/watch?v=ODSJsviD_SU (visited on 11/20/2022).

[55] *Agreement Concerning the International Carriage of Dangerous Goods by Road, Volume II*. Economic Commission for Europe Inland Transport Committee. New York and Geneva, 2020. URL: https://unece.org/sites/default/files/2021-01/ADR2021_Vol2e.pdf.

[56] *Digitraffic https://www.digitraffic.fi/en/road-traffic/*. https://www.digitraffic.fi/en/road-traffic/. Accessed: 2023-01-20.

[57] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. and Houlsby, N. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. DOI: 10.48550/ARXIV.2010.11929. URL: https://arxiv.org/abs/2010.11929.

[58] Wojke, N. and Bewley, A. Deep Cosine Metric Learning for Person Re-identification. (Mar. 2018). DOI: 10.1109/wacv.2018.00087. URL: https://doi.org/10.1109%2Fwacv.2018.00087.

[59] Xinchen, L., Wu, L., Huadong, M. and Huiyuan, F. Large-scale vehicle re-identification in urban surveillance videos. (2016).

[60] Zheng, L., Bie, Z., Sun, Y., Wang, J., Su, C., Wang, S. and Tian, Q. MARS: A Video Benchmark for Large-Scale Person Re-identification. (2016).

[61] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S. and Murphy, K. *Speed/accuracy trade-offs for modern convolutional object detectors*. 2016. DOI: 10.48550/ARXIV.1611.10012. URL: https://arxiv.org/abs/1611.10012.