Enhancing Airport Security with Computer Vision

A Comparative Study of Deep Learning Models for Efficient and Reliable Threat Detection in X-Ray Imagery of Luggage

Michiel de Ronde

Master Thesis Business Analytics

Enhancing Airport Security with Computer Vision

A Comparative Study of Deep Learning Models for Efficient and Reliable Threat Detection in X-Ray Imagery of luggage

Michiel de Ronde

Student number: 2643320

Vrije Universiteit Amsterdam

Faculty of Science

De Boelelaan 1081a

1081 HV Amsterdam

First Reader: dr. Anil Yaman

Second Reader: dr. Rene Bekker

Accenture

Data & AI - Engineering for AI

Fred. Roeskestraat 115

1076 EE Amsterdam

Client: Daniel Perez Jensen

Buddy: Roberto Navarro San Martin

October 2024

Amsterdam

Preface

This paper is written for the Master Project in Business Analytics at the Vrije Universiteit Amsterdam. It focuses on the use of computer vision to enhance the X-ray screening procedure of carry-on luggage at airports, a process that is currently inefficient, expensive, and inconsistent. By addressing this challenge, the research aims to improve the passenger security screening procedure at airports as a whole.

The research was conducted at Accenture, a global professional services company, and is a collaborative effort between the Vrije Universiteit Amsterdam and Accenture's Data & AI department (specifically the Engineering for AI team). It aims to combine academic expertise and real-world corporate context, allowing for a combination of theory and practice.

I would like to express my gratitude to my thesis supervisor, Dr. Anil Yaman, whose academic guidance and expertise were instrumental in shaping the structure and outline of this research. I would also like to extend my appreciation to Dr. Rene Bekker, my second reader, for his valuable support.
Finally, I would like to thank my Accenture supervisors, Daniel Perez Jensen and Roberto Navarro San Martin, for introducing me to a large corporate environment, providing valuable insights during my research, and helping me bridge the gap between academia and industry.

This research is dedicated to the ongoing efforts to make air travel safer and more efficient, and I hope it can serve as a stepping stone toward further advancements in airport security technology.

Abstract

The rapid growth in global air travel has significantly increased the demand for efficient and reliable airport security screening systems. Traditional methods for screening hand-luggage, which rely heavily on manual human inspection of X-ray images, often face challenges related to efficiency, consistency, and costs. This research aims to address these challenges by exploring the use of computer vision models for the detection of prohibited items in X-ray scans of hand-luggage.

During this study, the SSD, Faster R-CNN, and FCOS object detection models are evaluated using the open-source SIXRay dataset. The research focuses on comparing the performance of these models in terms of mean average precision and inference speed to identify a suitable approach for real-world deployment, and shows that the anchor-based two-stage Faster R-CNN model is best suited in this context. Furthermore, this research extends the evaluation of object detection models by introducing the background detection metric, which measures a model's ability to distinguish between positive and negative samples.

In addition, a selection of efficient classifiers is evaluated on the SIXRay dataset, outperforming the object detection models in the background detection task while being less computationally expensive. Because of this, a model pipeline is proposed consisting of an image classifier and an object detection model. The image classifier selects bags containing prohibited items and sends them to the object detection model, which does the localisation and classification of the object. Results showed that the proposed model pipeline has the ability to replace security operators in hand luggage X-Ray screening, making the current process less dependent on human operators, and thus cheaper, while at the same time increasing the efficiency, consistency, and safety of the security process.

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Over the last 50 years, the total number of air passengers has shown massive growth (Airlines for America 2024 [4]). During this period, civil aviation has evolved from a heavily regulated system exclusively available to the elite, to a much larger and more competitive global industry that is available to almost everyone in the western world.
It does not seem like this sector has reached its maximum. Gillen & Morris [14] state that the number of air passengers is predicted to grow at an average annual rate of between 4.2 and 4.7 percent through to 2033. That means that by 2030, approximately six billion passengers annually will use air transportation around the world. More importantly, all of them will require security screening at airports.

Baggage screening plays a central role within the transportation security domain. Identifying and detecting dangerous or prohibited objects in the hand luggage of passengers is a crucial factor in guaranteeing passenger safety and well-being. Because airports act as the connection between aircraft and land transportation, Barros & Tomber [5] state that airports represent the last opportunity to deny boarding to any person or object that poses a threat to the integrity of the aircraft and the people it carries. Because of this, airports are required by law to take on the responsibility of passenger and luggage security screening (Skorupski & Uchroński [49]).

Currently, hand luggage security checks predominantly rely on manual human inspection of

X-ray images. Petrozziello & Jordanov [41] state that a variety of factors make this a challenging process. Firstly, baggage often contains a wide range of unordered overlapping items varying in shape and size, giving this problem a complex context and increasing the difficulty of correctly recognising dangerous items. Additionally, a decision on the safety of the luggage has to be made quickly and consistently, adding time pressure to security operators. Finally, very few bags actually contain prohibited items, leaving room for bias and human error.

Furthermore, the current process is inefficient and expensive. Each piece of luggage must be checked separately by a team of professionally trained personnel, which is slow, costly, and requires a lot of manpower.
The success of airports like Schiphol is highly dependent on efficiently getting passengers through security, and not being able to do so could have major consequences. An example of this is the Schiphol security personnel shortage during 2022 (NOS, 2022 [58]), resulting in an extreme increase in passenger waiting time. This indicates the major disadvantage of being reliant on expert personnel and the importance of making security as efficient as possible.

Rapid advancements in deep learning and other novel computer vision algorithms provide possibilities to solve these problems. If done successfully, a large part of the baggage screening process could be improved, making it less dependent on human operators, cheaper, more efficient, and more reliable. As presented by Zhao et al. [63], computer vision is already extensively used in a wide range of industries. More specific examples include Shen et al. [48], who use a convolutional network to detect breast cancer on mammogram screenings. Or Kulkarni et al. [26], who use R-CNN to recognise traffic lights for self-driving cars. Although these examples differ from the problem at hand, they demonstrate the suitability of computer vision for highly complex and intricate real-world problems.

Within the transportation security domain, major airports such as Schiphol have recently started introducing smart scanners for on-person item checks that rely heavily on computer vision (Algemeen Dagblad, 2024 [8]), indicating the relevance and potential of leveraging computer vision in this industry. However, the specific case of detecting prohibited items in X-ray images of hand luggage seems to have fallen behind in both literature and practice. The problem with current research on this subject is two-fold. Firstly, as highlighted by Mery et al. [33], almost all research is conducted using private or classified datasets. As a result, (numerical) findings cannot or only partially be reported and much of the literature becomes not reproducible. This makes reliable comparisons and future research possibilities scarce. Furthermore, new models and frameworks that are proposed often rely on new or unusual scanning procedures. This means airports likely have to make major and expensive changes in their security infrastructure before they can adopt these new methods. As this is both expensive and high risk, the chance of adoption becomes small.

That is why this research aims to compare multiple computer vision models in their ability to detect contraband in X-ray scans of hand luggage on a reliable open source dataset without the need for airports to make expensive or major changes. All in all, this paper aims to present an applied theoretical framework/proof of concept for the use of computer vision in airport security, and by doing so, achieve the research goal and answer the research question below:

Research Goal: *Develop and compare computer vision models that detect prohibited items in X-ray images of hand luggage without the need for airports to make expensive or major changes using a reliable open-source dataset*

Research Question: *What performance level can computer vision models obtain in the detection of prohibited items in X-ray images of hand luggage and how do different methods compare?*

The paper will continue with a more extensive problem description in Chapter 2, discussing the current passenger security screening procedure and its problems both in a practical and business context. Chapter 3 will present the relevant background and related work. This will be followed by the outline of the main research in the methodology (Chapter 4), experimental setup (Chapter 5), and results (Chapter 6). Finally, the paper will discuss the presented research and future research possibilities in Chapter 7, followed by a final conclusion in Chapter 8.

Chapter 2

Problem Description and Business Context

The screening of passengers and their baggage is not a recent innovation. Barros & Tomber [5] note that passenger and baggage screening has been widely used in both North America and Europe since the 1960s and 1970s. The complete screening procedure consists of two main parts: checked baggage screening, and the screening of passengers and their hand-luggage. Checked baggage screening, as shown in Figures 2.1 and 2.2, involves the process of inspecting all baggage that is checked in by passengers, i.e. baggage that is stored in the plane's cargo hold and inaccessible to passengers throughout the flight. As this part of the screening process is independent from passenger and hand-luggage screening, it falls outside the scope of this research and will not be further analysed in this chapter.

The passenger and hand-luggage screening procedure consists of checking passengers along with their hand-luggage. The purpose of these screenings has always been the same: to catch and confiscate objects that could be used as a weapon or otherwise pose a threat to security.

In order to be able to improve the passenger and hand-luggage screening procedure, the current process must first be understood and analysed. That is why this chapter will discuss the current state of passenger and hand-luggage security screening (Section 2.1). Additionally, the current screening process will be evaluated in order to highlight bottlenecks and determine its shortcomings (Section 2.2). The security screening process of passengers and their hand luggage will be analysed and evaluated on two main parts: safety and efficiency. In this context, safety refers to the actual ability of the system to catch and confiscate contraband, while efficiency relates to how well the system is able to handle both small and large passenger loads. Finally, this chapter will consider the screening process in a business context and state why improvement on these topics is in the interest of airports and airlines (Section 2.3).

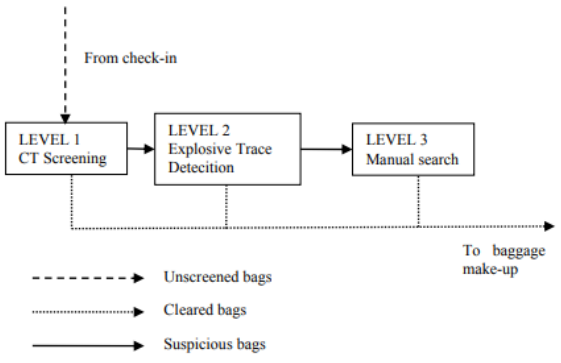

Figure 2.1: Checked baggage screening model used in the US.

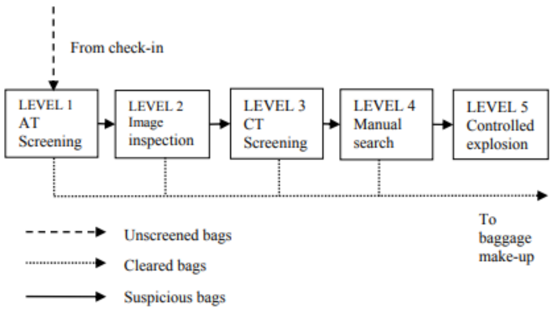

Figure 2.2: Checked baggage screening model used in Europe and Canada

2.1 Current Passenger Security Screening Process

The screening of passengers and their hand-baggage is performed at security checkpoints. Barros & Tomber [5] state that in the last two decades the use of security channels has become the standard practice for passenger security checkpoints in most airports. Upon arrival at the security checkpoint, passengers are divided into separate channels equipped with a body screening machine and a baggage screening machine. Usually, body screening is done via some kind of metal detector, while an X-ray machine is used for baggage screening.

The security checkpoints are located at entrance points to the boarding gates. By doing so, boarding gates become exclusively accessible to individuals who have undergone security screening. The security checkpoints thus separate the airport into an ’unsecured’ main area and a ’secured’ area where all boarding gates are located. Figure 2.3 illustrates a conceptual example of the main unsecured building, the secured boarding area, and the placement of the security checkpoint between the two.

Graves et al. [18] describe the screening of passengers and their hand-baggage at security

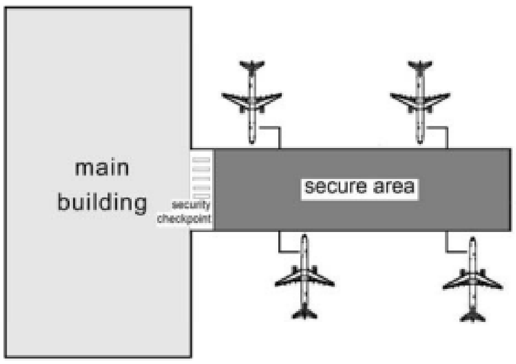

Figure 2.3: Security checkpoint placement between main building and boarding area.

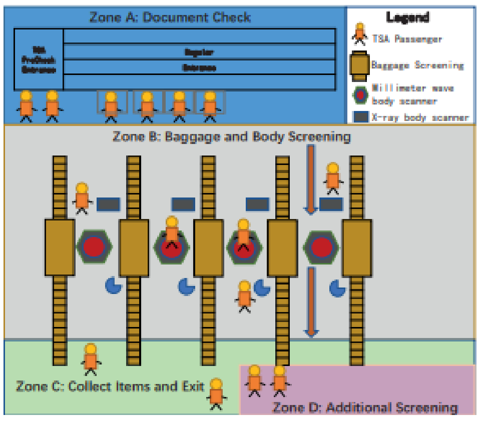
Zone A: Document Check
Regular
Entrance
Legend
TSA Passenger
Baggage Screening
Millimeter wave body scanner
X-ray body scanner
Zone B: Baggage and Body Screening
Zone C: Collect Items and Exit
Zone D: Additional Screening

Figure 2.4: Security checkpoint layout.

checkpoints in great detail. The screening process is divided into five main parts: X-ray screening of property, walk-through metal detector screening of individuals, pat-down screening of individuals, physical search of property, and trace detection of explosives. Individuals are usually only required to submit their baggage for X-ray screening and walk through a metal detector. However, if an alarm for a particular individual or piece of baggage is raised at either part of the screening, the individual and/or their possessions may receive additional screening. Furthermore, individuals and/or their baggage can also be selected for further evaluation due to suspicious behaviour or random selection.

The basic procedure for screening individuals and their baggage at the security checkpoints is presented below:

1. Each individual places small items located on their person in a tray and then places the tray and any hand-luggage items on the X-ray conveyor belt. This process is supervised by security staff, who will tell passengers which items to remove from the baggage and ensure that items are placed appropriately on the conveyor belt.

2. Individuals then enter the walk-through metal detector under the supervision of another security staff member while their baggage passes into the X-ray machine for evaluation.

3. If the metal detector activates an alarm, individuals need to remove the (forgotten) items that raised the alarm. The offending items are sent through the X-ray machine while the

individual may be subjected to a pat-down search and/or another walk through the metal detector.

4. Meanwhile, a specially trained security operator (possibly two) watches the display screen of the X-ray machine and decides if there are any prohibited items displayed in the X-ray images of the baggage.

5. The security operator who evaluates the X-ray images is in control of the conveyor belt. This operator has the power to stop or slow down the movement of containers for a more detailed examination if necessary. Besides this, the operator is also able to manipulate settings on the X-ray machine to enhance the presented image if necessary. Additionally, if the operator deems it necessary, any object may be resubmitted through the machine at a different orientation for a second assessment.

6. When the operator(s) is certain that there is no contraband within the baggage, the container will be allowed to move from the conveyor belt back to the corresponding individual. After all items are collected, the individual can leave the security checkpoint.

7. However, if the operator decides that there is a suspicious object present, he/she will hand over control of the container to other security staff for follow-up action. This will involve both the object in question as well as the individual who owns it. follow-up actions can consist of questioning, physical search, and/or explosive trace detection.

8. If further evaluation indicates that no prohibited items are present, the individual can collect their items and leave the security checkpoint. If prohibited items are found or suspicion still remains, appropriate security measures need to be taken by security staff.

2.2 Passenger Security Screening Evaluation

The above described security checkpoints operate under a lot of pressure. Literature like Kierzkowski & Kisiel [22] and Knol et al. [24] refer to security checkpoints as a key element in an airport and state that they should be both safe and efficient. Safety is important to ensure the well-being of passengers and efficiency to avoid bottlenecks in passenger flow. This is challenging as one usually comes at the cost of the other. That is why this section will evaluate the current security checkpoints, as described above, on exactly these two aspects.

While literature like Michel et al. [37] and Kierzkowski & Kisiel [21] propose frameworks for airport security evaluation, they do not report actual numerical results due to safety concerns or a lack of data. However, as Petrozziello & Jordanov [41] state that the safety performance of security checkpoints is highly dictated by the inspection of X-ray images, the assumption is made to use the performance of this part of the checkpoint as an indication of the safety performance of the whole system. Meuter & Lacherez [35] investigated the performance of security operators in detecting prohibited items in X-ray images at a major Australian international airport. Performance was measured in terms of the percentage of correct detections (measured as accuracy) by using Fictitious Threat Items (FTI's). The authors measured a mean accuracy of 93.5%, with a standard deviation of 4.27%. Although the mean accuracy seems high, there was considerable variation in overall accuracy, with some operators scoring considerably worse, between 80% and 90%, and one operator even scoring as low as 71%. This highlights a significant issue in the safety of current security checkpoints: The performance of individual X-ray security operators substantially impacts the overall security.

Other research such as that presented by Graves et al. [18] and Kirschenbaum [23] confirms the claim that the performance of airport security checkpoints is strongly influenced by the individual performance of security operators responsible for the inspection of X-ray images.
Graves et al. [18], for example, state that different security operators can show different levels of bias, depending on their training. They also suggest that security operators have different thresholds that must be surpassed before something is identified as a threat. Many factors can influence this, including time pressure, accountability, expectations, and emotions surrounding the task. This is confirmed by Knol et al. [24], who mention that "10% of security personnel exceeds or bends rules when the situation calls for it", and “12% of security personnel states that breaking (security) protocol is sometimes necessary to get the job done". In addition, it has been found that human operators often base their decision making on past experiences, which

differ between operators, more than on logic or rationality. This all leads to a large variety in the performance of security operators, and thus security checkpoint safety.

Besides differences between security operators, the intraindividual performance of security operators also seems to vary a lot. In their research, Meuter & Lacherez [35] saw that during shifts which were characterized by greater busyness (greater number of bags seen per minute), performance greatly deteriorated as the shift continued. Performance declined only slightly during the very first trials, and then began to decline at an increasing rate. Notably, performance began to drop as early as 10 minutes into the shift, and dropped rapidly after this time.

But inconsistency does not seem to be the only problem with the manual inspection of X-ray images. Research, as presented by Barros & Tomber [5] and Su et al. [52], indicate the X-ray screening of passenger carry-ons to be the main bottleneck in the security checkpoint system. Using a simulation model, Barros & Tomber [5] analysed the passenger and hand-luggage screening procedure at Seattle-Tacoma (Sea-Tac) International Airport. Results of the analysis proved that it is possible to drastically reduce waiting times with changes in the X-ray screening procedures. Additionally, the authors state that "Clearly, the X-ray is a significant bottleneck in the screening process.".

This same conclusion is reached by Su et al. [52], who researched the airport security inspection process based on both a retention and regression model. They describe the security checkpoints by dividing them into four zones: document checking (zone A), items checking (zone B), items collecting (zone C), and additional screening (zone D). A visual representation of these zones can be found in Figure 2.4. When applying the retention model, it was found that the retention index of zone B was significantly larger than that of zone A and zone C, indicating that zone B is more likely to become the bottleneck area in a security checkpoint. Furthermore, a regression model was used to model waiting time. From the established waiting time model, passenger waiting time was mainly related to the security process in zone B, thus again indicating a bottleneck at this part of the checkpoint. The authors conclude by stating that "The results show that the scan time in items checking zone (zone B) has the greatest impact on passenger waiting time, zone B is the bottleneck area of airport security inspection".

2.3 Business Context

The performance of security checkpoints has a large influence on both airport and airline income/costs and should therefore also be considered in a business context.
Gillen & Morris [14] estimate that in 2002, total European expenditures on aviation security were around €2.8 billion. With an estimated total spending on aviation security of €5.7 billion in 2011, this has more than doubled in less than 10 years. Another indication of this immense growth in security costs is the US government funding of the Transportation Security Agency (TSA), which has increased significantly since its inception, growing from $2.2 billion in 2002 to almost $8 billion in 2013. Taking this massive growth into account, the authors state that unless significant changes are made, the total costs of current aviation security are likely to reach unsustainable levels over the next 15-20 years.
Part of these massive costs are related to security checkpoints. Barros & Tomber [5] mention that massive queues and long waiting times at security checkpoints have created the need for expansion of the inspection areas, driving up costs. But the main driver behind the costs of security checkpoints is the personnel. Because current security checkpoints are highly dependent on specialised personnel with all the necessary qualifications and certificates to perform their duties, they are expensive. Skorupski & Uchroński [49] note that with simple calculations, it can be stated that the minimum staffing of a single security channel requires 4-5 employees in 24 hours. This is a factor that generates huge costs for the total security checkpoint (which consists of multiple security channels). To ensure continued operation of a single security channel, 12-15 workers have to be employed. For an airport with around 3 million passengers per year, the cost of security (equipment and staff) can reach about € 4 million per year.

Besides being a significant factor when it comes to airport costs, security checkpoints also play a major role in airport income generation and customer satisfaction. As mentioned in Section 2.2, the current passenger security screening process has some major problems, most of which have a strong effect on customer satisfaction. This has been proven by studies from the likes of Pande & Hazare [38], who show that there is a strong link between customer satisfaction and improved security measures. Additionally, Wei & Cheng [59] statistically prove that airports with faster security are preferred by travellers. As a result, the same research shows that airports with long queues and waiting times are actively avoided by travellers. Based on these studies, it can be pointed out that more efficient security screening will attract more passengers, giving airports a competitive advantage and the opportunity to increase income.

Taking Sections 2.1, 2.2, and 2.3 into consideration, it can be concluded that the current passenger security screening process does not perform well. The system is not only inconsistent, unsafe, and inefficient, it is also expensive because of its massive reliance on human operators. When inspecting the process more closely, it can be seen that the X-ray screening of hand-luggage appears to be the underlying issue. Enhancing this specific part of the screening process has the potential to result in major improvements in the security system, and thus for airports as a whole. The use of computer vision could offer solutions to solve the underlying problems. That is why this research experiments with computer vision models trained on baggage X-ray imagery. By doing so, it aims to enhance the described security process, making it more efficient, more consistent, and cheaper, while at the same time increasing safety and providing opportunities to increase income.

CHAPTER 3

Background and Related Work

This chapter will discuss the relevant literature and concepts necessary to fully understand and reproduce the presented research. Firstly, it will briefly introduce the basic building blocks of neural networks (Section 3.1), a central concept in this research. Secondly, it will present the use of neural networks in two popular applications: classification (Section 3.2) and object detection (Section 3.3). Finally, this chapter will discuss literature related to this research (Section 3.4).

3.1 Neural Networks

3.1.1 Basic Neural Networks

The idea behind neural networks was first introduced in the early 1940s. Inspired by the human brain (which contains neurons), McCulloch et al. [32] presented a mathematical model for neurons when aiming to create early versions of artificial intelligence. The idea of a neuron needed to be radically simplified, and doing so yielded one of the first successful machine learning systems: the perceptron as introduced by Rosenblatt [45]. A perceptron takes multiple input variables (or nodes), that each get multiplied with a corresponding weight. The weighted inputs, along with a bias parameter, are then summed to provide the perceptron's output. The bias parameter is typically represented as a distinct input node, referred to as the bias node, which consistently holds a value of 1. The weights and bias are the parameters of the model, these can be tuned in order to get the desired output. A general example of a perceptron can be seen in Figure 3.1. As can be seen, this is simply a linear regression model drawn as a network. Because of this, the presented perceptron structure proves to be an overly simplistic abstraction. Using the output of a perceptron as the input for another (composing perceptrons) does not enhance their computational power. This limitation arises because perceptrons are inherently linear functions. Consequently, the composition of linear functions will always result in another linear function, no matter how the perceptrons are chained together. These models are thus incapable of learning non-linear functions.

In later research, Rosenblatt [44] deals with this problem by applying a non-linear function to each neuron, the activation function. This scalar function, which maps a single numerical input to a single numerical output, is applied to the perceptron's output after the aggregation of all weighted inputs and results in the abbility to learn non-linearities. Some popular activation functions are the Sigmoid, ReLu, and Softmax function. The Sigmoid nonlinearity, as first presented by Lettvin et al. [28], maps input values to a range between 0 and 1. The ReLu function (Glorot et al. [16]) outputs the input directly if it is positive, and otherwise outputs zero. Finally, the Softmax function (Rumelhart et al. [47]) converts a vector of raw scores (logits) into a probability distribution. Using no activation function is also called a linear activation.

By incorporating these non-linear functions, individual neurons can be organised into neural networks (i.e. any arrangement of perceptrons and nonlinearities). An example of such a neural network is the multilayer perceptron as shown in Figure 3.2. Note that every orange and blue line in this figure represents one parameter of the model. All the horizontal layers of nodes between the input and output layer (in this case 1) are called hidden layers. The more hidden layers a model has, the deeper a model is, resulting in more parameters. The sequential processing of input data through the input layer and hidden layers to produce an output is called a forward pass.

3.1.2 Neural Network Training

The goal of training a neural network is to optimise its weights so that it can accurately map input data to the desired output. This is done using the backpropagation algorithm, as first popularised

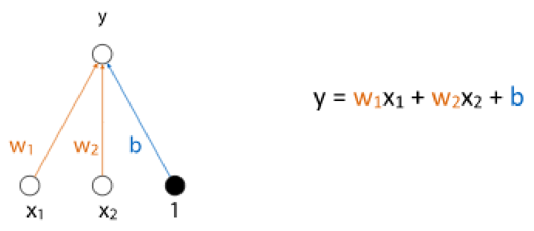

Figure 3.1: Example of a perceptron.

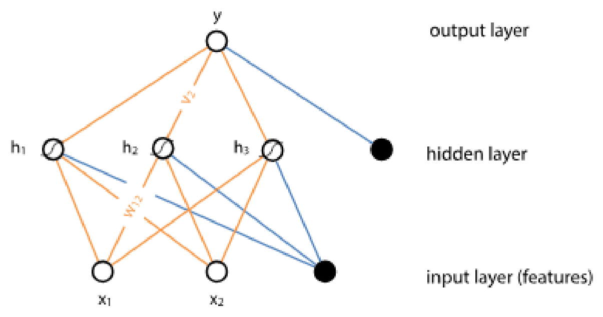

Figure 3.2: Example of a multilayer perceptron.

by Rumelhart et al. [46]. The process involves three primary stages: the forward pass, the loss calculation, and the backward pass. As mentioned in Section 3.1.1, a forward pass propagates the input data through the network to produce an output. During training, this output is used to compute the loss.

The loss reflects how the model performs on a given set of input data and is calculated by the loss function. In his book, Crowley [10] states that by computing the difference between the predicted output of the model and the actual target values, the loss serves as the objective that a neural network aims to minimise during training. In the context of supervised learning, the loss function is evaluated at the output layer, which compares the predicted values with the ground truth (which is known). The choice of the loss function can significantly impact the final performance of neural networks. It is thus important that a correct loss function is chosen that aligns with the task and the nature of the output, guiding the neural network to optimise its parameters effectively and generalise well to unseen data.

The computed loss is then used during the backward pass. In the backward pass, the backpropagation algorithm computes how the weights should be adjusted to reduce the loss (and thus minimise the error). As mentioned by Gershenson [13], backpropagation is based on the principle of gradient descent. Gradients of the loss function with respect to each weight are calculated by propagating errors backward through the network. The weights of a neural network are then optimised by moving in the direction of the steepest descent during each training step. In most modern deep learning frameworks, this gradient computation is done using automatic differentiation (autograd). Autograd, the core mathematical framework of which was first introduced by Linnainmaa [29] and later applied to neural networks by Rumelhart et al. [46], efficiently computes the gradients for all model parameters by building a computational graph during the forward pass. This graph is then traversed backward during the backward pass. Each operation in the network has both a standard forward function, responsible for computing the output, and a backward function. This backward function is used for calculating the gradients during backpropagation. Each layer has its own backward function, which defines how gradients from the subsequent layer are combined with local activations and weights to compute the gradient for that layer. Backpropagation, combined with automatic differentiation, is essential for training deep neural networks, enabling them to model complex, non-linear relationships in data.

3.1.3 Convolutional Neural Networks

Sections 3.1.1 and 3.1.2 explain the basic concepts of neural networks. Over the last couple of years, these concepts have paved the way for groundbreaking innovation. So much so that Alzubaidi et al. [3] call neural networks the "gold standard in machine learning". One of these major innovations has been the introduction of convolutional neural networks (CNN). A CNN is a specialised type of neural network that is mainly used to process structured grid-like data like images. CNNs have become fundamental in tasks related to image recognition, classification, and other computer vision applications due to their ability to capture spatial hierarchies in data. The CNN was first introduced by LeCun et al. [27] who in their research described the LeNet-5 architecture. LeNet-5 was used for digit recognition and applied convolutional layers for the first time. It became one of the pioneering CNN architectures and laid the foundation for modern deep learning approaches in computer vision.

In their work, Alzubaidi et al. [3] describe CNNs in great detail. In addition to the standard components of a neural network, as explained in the previous sections, the basic structure of CNNs consists of three main types of layers: convolutional layers, pooling layers, and fully connected layers.

1. The convolutional layer is the main building block of a CNN. In this layer, learnable filters (or kernels) slide across the input data, processing a subset of the input data at each position. An example of such an operation is shown in Figure 3.3. This process outputs

what are called feature maps, which highlight local patterns in the data. If for example an image is used as input data these could be things such as edges, textures, or shapes. By stacking multiple convolutional layers (using the output of a layer as the input of the next layer), CNNs can learn complex and hierarchical features. When stacking convolutional layers, lower layers usually detect the simpler patterns while higher layers capture more complex details.

2. Pooling layers are often used after convolutional layers and reduce the spatial dimensions of the produced feature maps. Just as with convolutional layers, pooling layers process a subset of the input data at a time. This approach shrinks large-size feature maps to create smaller feature maps while maintaining the majority of the dominant information. This decreases the computational load and helps to prevent overfitting. The most familiar and frequently used pooling methods are max, min, and average pooling. Max/min pooling retains the maximum/minimum value in each patch of the feature map, thus preserving critical features while discarding irrelevant ones. Average pooling takes the average of each data subset. Sometimes, the average is taken over the whole input, this is called global average pooling. Examples of these pooling methods are shown in Figure 3.4.

3. After a series of convolutional and pooling layers, the output is typically flattened and passed through one or more fully connected (FC) layers. Usually, this layer is located at the end of the CNN architecture. As the name already mentions these layers are fully connected, meaning that each neuron is connected to all neurons of the previous layer. The input of the FC layer comes from the last pooling or convolutional layer. This input is in the form of a vector, which is created from the feature maps after fattening. The output of the FC layer represents the final CNN output.

Key to the success of CNNs is their ability to maintain the spatial structure of the input, thus making them particularly effective for image data. Goodfellow et al. [17] identify three key benefits of CNNs: equivalent representations, sparse interactions, and parameter sharing. Unlike conventional fully connected (FC) networks, shared weights and local connections in the CNN are employed to make full use of 2D input-data structures. This operation utilises an extremely

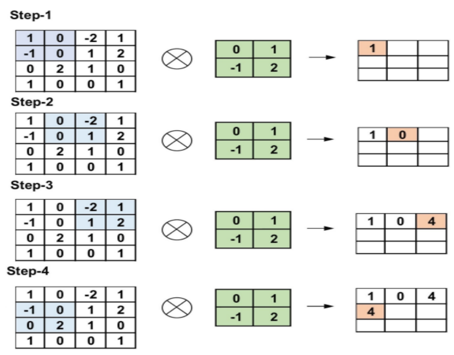

Figure 3.3: Example of a convolutional layer in a CNN.

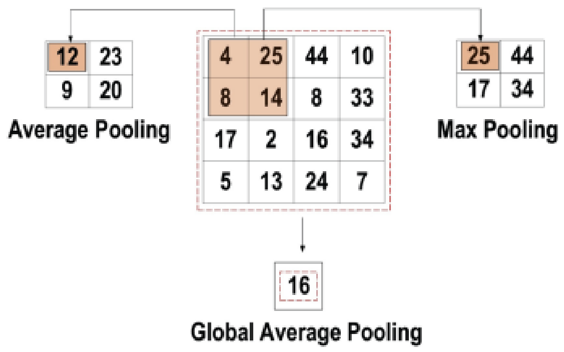

Figure 3.4: Example of max and (global) average pooling.

small number of parameters, which both simplifies the training process and speeds up the network.

Due to their hierarchical nature, CNNs have achieved state-of-the-art performance in various domains of computer vision like classification and object detection.

3.2 Image Classification

Classification is a popular task in machine learning. The goal of this task is to assign predefined labels to input data based on its features. Traditional classification algorithms, such as random forest (RF), support vector machines (SVMs), and k-nearest neighbours (k-NN), usually work by using predefined rules or distance metrics to separate data points into distinct classes. As Chen et al. [9] point out, these models often depend heavily on feature engineering, especially when dealing with high-dimensional datasets. That means that features have to be manually selected and extracted from the data for the model to perform well. This process can be challenging for high-dimensional or unstructured data such as images, text, and video, making these traditional approaches unsuitable for computer vision tasks.

One such example of a computer vision task is image classification, which seeks to differentiate between distinct classes of objects based on various properties reflected in an input image. In other words, a computer can identify the class to which the objects in an image or video belong. Zhao et al. [62] mention that traditional classification algorithms, as mentioned previously, can achieve the expected results in simple (image) classification tasks. However, their performance in complex classification tasks is not satisfactory. The authors also mention that the CNN framework offers solutions. By using convolution kernels, CNN's are able to extract features from the original input and automatically learn feature representations from a large amount of sample data. This leads to CNN models having stronger generalisation abilities compared to conventional image classification algorithms that manually extract features.

The main process of image classification using a CNN includes preprocessing the original image, extracting image features, and classifying the image using a classifier, in which the extraction of image features plays a pivotal role.

The following sections describe some popular CNN classifiers in more detail, discussing the main ideas behind the models and most important insights from accompanied literature.

3.2.1 ResNet

The Residual Network (ResNet) is a CNN framework that aims to create deep neural networks with high accuracy. Before ResNet was introduced by He et al. [19], neural networks often experienced the degradation problem, where accuracy increased until a certain point and then rapidly declined as the depth of the network increases.

Inspired by the highway network concept, as introduced by Srivastava et al. [51], ResNet resolves this problem by using stacked residual blocks (Figure 3.5). These blocks are composed of not only standard convolutional layers, but also contain a direct connection between its input and output called the shortcut connection. As can be seen, the output of the convolutional layers is called $F(x)$. To compute the final output of the residual block, the output of these layers ($F(x)$) is added to the output of the shortcut connection (x), giving $F(x)+x$ as the final result.

These residual blocks are the core idea behind ResNet. Instead of directly mapping the input x to the desired output $H(x)$, ResNet models this relationship as $H(x) = F(x) + x$. Learning $F(x)$ is simpler and easier to optimise compared to learning the original mapping $H(x)$. The shortcut connections allow gradients to flow directly through the network and enable the exchange of features between different layers, preventing the vanashing gradient problem and making it easier to train deep networks.

Despite the increased depth, ResNet maintains low complexity, making it computationally efficient while achieving state-of-the-art performance in image classification and other computer vision tasks. In their original work, He et al. [19] demonstrated that ResNet significantly outperforms plain networks of similar depth, showing that residual learning enables much deeper architectures to be trained successfully without degradation in performance.

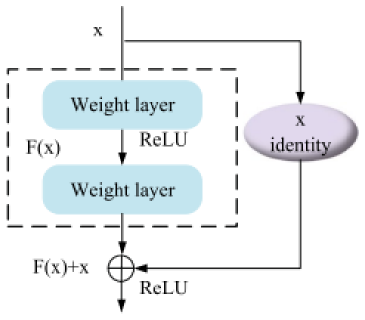

Figure 3.5: Example of a residual block.

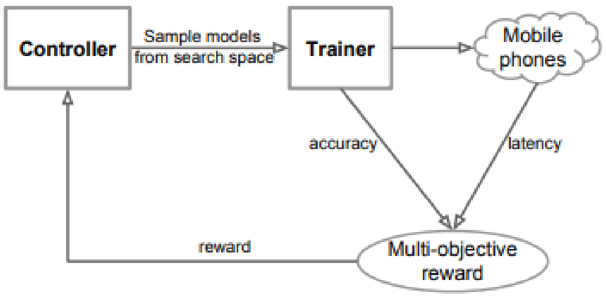

Figure 3.6: MNAS searching algorithm.

3.2.2 MNASNet

MNASNet is a CNN architecture designed mainly for mobile devices. As mobile models need to be both small/fast and yet still accurate, the design process can be challenging. In their research, Tan et al. [54] propose an automated mobile neural architecture search (MNAS) approach to design mobile models, which is then used to create the MNASNet model.

The MNAS approach formulates the design problem as a multi-objective search, explicitly incorporating model latency into the main objective. By doing this, the search can identify a model that achieves a good trade-off between accuracy and latency. To achieve this, a customised weighted product method is used to approximate Pareto optimal solutions. The optimisation goal is defined in Equation 3.1, where w is the weight factor as defined in Equation 3.2 and α, β are application-specific constants.

In addition to the objective function, a well-defined search space is extremely important for neural architecture search. In contrast to previous approaches, a novel factorised hierarchical search space is introduced that factorises a CNN model into unique blocks and then searches for the operations and connections per block separately. This encourages different layer architectures in different blocks, which is crucial for improving computational efficiency while maintaining high accuracy. This design approach contrasts with previous architectures, where the same types of cells are repeatedly stacked, which can limit the diversity and performance of the network.

$$\max_{m} ACC(m) \times \left[\frac{LAT(m)}{T}\right]^{w} \tag{3.1}$$

$$w = \begin{cases} \alpha, & \text{if } LAT(m) \leq T \\ \beta, & \text{otherwise} \end{cases} \tag{3.2}$$

The search space is explored using a reinforcement learning algorithm (Figure 3.6) that consists of three main parts: a controller, trainer, and inference engine. The controller generates potential CNN architectures by outputting a sequence of actions (tokens) that define a CNN structure. Once a CNN structure is generated, it is trained to evaluate its accuracy on the target task. The trained model's inference latency is then measured on a real mobile device. This sample-evaluate-update loop continues iteratively, where the controller updates its parameters to maximise the reward function. This function is presented in Equation 3.3, where m is a sampled model determined by action $a_{1:T}$, and $R(m)$ is the objective value defined by Equation 3.1.

$$J = E_{P(a_{1:T};\theta)}[R(m)] \tag{3.3}$$

Experimental results show that the approach consistently outperforms state-of-the-art mobile CNN models across multiple vision tasks on both accuracy and efficiency. This ability to tailor architectures for real-world latency constraints while maintaining competitive performance makes MNASNet highly suitable for deployment in resource-constrained environments.

3.2.3 EfficientNet

EfficientNet is a type of CNN architecture that is designed to be more efficient compared to other CNN's while maintaining competitive performance. Traditionally, the process of improving CNN performance involved individually increasing the depth (number of layers), width (number of nodes per layer), or input resolution of the model. In their research, Tan & Le [53] found that carefully balancing network depth, width, and resolution can lead to improved results. Based on this observation, a new scaling method is proposed that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective compound coefficient. This balanced scaling allows the network to capture more complex patterns without the need of manual tuning for each dimension. The mathematical definition is shown in Equation 3.4, where α, β, γ are constants that can be determined by a small grid search. Intuitively, ϕ is a user-specified coefficient that controls how many more resources are available for model scaling, while α, β, γ specify how to assign these extra resources to network width, depth, and resolution respectively.

$$\begin{aligned} \text{depth: } d &= \alpha^{\phi} \\ \text{width: } w &= \beta^{\phi} \\ \text{resolution: } r &= \gamma^{\phi} \\ \text{s.t. } \alpha \cdot \beta^{2} \cdot \gamma^{2} &\approx 2 \\ \alpha \geq 1, \beta \geq 1, \gamma &\geq 1 \end{aligned} \tag{3.4}$$

The EfficientNet architecture is based on a relatively small baseline network called EfficientNet-B0, which was discovered using neural architecture search (NAS) techniques. The compound scaling approach is then applied to this baseline to create a family of models, from EfficientNet-B1 to EfficientNet-B7. Each successive model in the family increases the depth, width, and resolution in a balanced manner, resulting in significant improvements in accuracy with fewer parameters and lower computational costs compared to traditional models.
In particular, EfficientNet-B7 achieves state-of-the-art performance while being 8.4 times smaller and 6.1 times faster on inference than previous models. The EfficientNet models also demonstrate excellent transferability, achieving state-of-the-art results on multiple transfer learning datasets, further highlighting their versatility and efficiency.

3.2.4 RegNet

RegNet is a family of convolutional neural networks designed to provide efficient model scaling while maintaining simplicity and flexibility. Introduced by Radosavovic et al. [42], the authors aim to design better networks for visual recognition. Rather than designing or searching for a single best model under specific settings, the behaviour of populations of models was studied. By doing this, general design principles that can be applied to improve an entire model population can be discovered.

RegNet models consist of stages, each composed of multiple residual blocks with increasing widths. The design of these stages follows a linear parameterisation that allows for scalable model construction. Using this structure, RegNet models achieve a high concentration of top-performing networks within the design space.

The core concept behind RegNet is that the widths of good neural network layers can be described by a parameterisation method. The idea behind this parameterisation is to model the block widths of a network linearly and then quantise them into discrete stages. First, a linear parameterisation for block widths is introduced. This means that the width of each layer of a certain block j, denoted as U_j, is calculated using a linear function (Equation 3.5). As can be seen, this function depends on three parameters: depth d, initial width $w_0 > 0$, and slope $w_a > 0$, and generates a different block width u_j for each block $j < d$.

$$u_j = w_0 + w_a \cdot j \quad \text{for} \quad 0 \leq j < d \tag{3.5}$$

To make these widths practical, the additional parameter w_m is introduced. By using u_j as calculated in Equation 3.5, s_j is computed for each block j such that Equation 3.6 holds. s_j is then rounded (denoted by $\lfloor s_j \rceil$) and used to compute the quantised per-block widths w_j using Equation 3.7.

$$u_j = w_0 \cdot w_m^{s_j} \tag{3.6}$$

$$w_j = w_0 \cdot w_m^{\lfloor s_j \rceil} \tag{3.7}$$

The RegNet family has been shown to outperform other state-of-the-art models in terms of computational efficiency and accuracy. RegNet models are particularly well suited for deployment on GPUs, where they demonstrate up to 5× faster inference times compared to other models while maintaining comparable or better performance in visual recognition tasks. This ability to generate simple, regular models that generalise well across various settings marks RegNet as a significant advancement in neural network design.

3.3 Object Detection

Besides image classification, another popular computer vision task is object detection. As the name suggests, object detection involves the process of detecting and classifying objects within an input image. This process typically consists of two steps: localisation and classification. First, the model tries to localise objects of interest in the input image. The found objects are then usually highlighted by a predicted bounding box. Secondly, the objects are sorted into several predefined classes. Compared to image classification, object detection focuses more on local regions of an image and specific sets of object classes.

Before the wide adoption of CNN's and other deep learning frameworks, object detection algorithms relied mainly on the traditional sliding-window approach and were designed manually. A sliding window traversed the input image and used feature descriptors such as Haar (Papageorgiou et al. [39]), Sift (Lowe [31]), or Surf (Bay et al. [6]) to extract features. These features are then used to train a separate shallow classifier, one for each class of target objects. However, Zhao et al. [62] mention that the traditional object detection approach suffers from a lack of robustness, poor generalisation, and low detection accuracy. The authors also mention that, in response to the problem of manual parameter tuning of traditional object detection algorithms, the research boom in deep networks has brought new opportunities for the development of object detection. Compared with traditional object detection algorithms, deep CNN's can automatically learn feature representations of parameters from massive data sets and do not require additional training of classifiers, which greatly improves the efficiency of the feature learning process.

This section will introduce three important object detection models, briefly discussing their general outline and how they work.

3.3.1 Single Shot Multibox Detector (SSD)

The Single Shot MultiBox Detector (SSD) is a convolutional neural network architecture designed for real-time object detection. As presented by liu et al. [30], the SSD is an anchor-based single-stage object detector that consists of four key elements: multi-scale feature map generation, default boxes and aspect ratios, convolutional predictors, and non-maximum supression (NMS).

The framework of the SSD model (Figure 1 in the Appendix) starts with a preexisting image classification base network that provides lower-level feature maps. In the original work, the VGG16 model is used as the base network, but Liu et al. [30] state that other similar networks should work as well. To this base network, additional convolutional layers are added. These layers progressively decrease in size, resulting in multiple feature maps of different sizes at different layers. This allows the model to detect objects at multiple scales and handle objects of different sizes. Small convolutional filters are then applied to the generated feature maps to predict object classes and bounding boxes. For each feature map, the filters output a class score (for object classification) and a bounding box (for localisation).

To make these predictions, the SSD model uses a number of predefined bounding boxes at each feature map. By using different aspect ratios and scales for the default bounding boxes at each feature map location, the model is able to cover a wide variety of object shapes. After the network produces its classification scores and bounding boxes, a non-maximum suppression (NMS) step is applied to eliminate redundant overlapping boxes and keep only the most confident predictions.

SSD's efficiency and accuracy stem from its use of default boxes across multiple scales, its avoidance of region proposal generation, and its direct prediction of object categories and bounding box locations. These characteristics make it one of the leading architectures for real-time object detection tasks.

3.3.2 Faster R-CNN

Faster R-CNN is a widely-used object detection framework that integrates region proposal generation and object detection in a unified architecture. It builds on the Fast R-CNN framework, improving both the speed and accuracy of object detection by introducing a Region Proposal Network (RPN). Just like the SSD model, the Faster R-CNN model uses an anchor-based approach. However, due to the region proposals, the Faster R-CNN is defined as a two-stage object detection model instead of single-stage like the SSD.

Fast R-CNN, as introduced by Girshick [15], takes the entire input image and a set of regions of interest (RoIs) as input. The image is first processed using a preexisting image classification base network to produce a convolutional feature map. For each region of interest, a RoI pooling layer extracts a fixed-size feature map from the convolutional feature map. The RoI pooling layer uses max-pooling to convert any sized RoI into a fixed-sized feature map. This makes it possible to use the same fully connected layers for all interest regions, regardless of their size or shape. The fixed-size feature maps generated by the RoI pooling layer are then passed through a series of fully connected layers before arriving at two output layers: one for classification and one for the prediction of the bounding boxes. Classification is done by a softmax layer that classifies the object into one of the predefined object classes. Bounding box predictions are made by refining the given RoIs. For each RoI containing an object, the bounding box prediction layer outputs the predicted offsets needed to refine the RoI bounding box around the actual object.

Faster R-CNN, as presented by Ren et al. [43] builds on the Fast R-CNN model by introducing a Region Proposal Network (RPN). This network handles the task of generating regions of interest, eliminating the need for separate algorithms for this specific task. The RPN is a fully convolutional network that shares convolutional layers with the Fast R-CNN object detection network as described above. By using a sliding window approach, it generates RoIs directly from the feature maps produced by the base network. The RPN generates multiple proposals at each spatial location, using predefined anchor boxes with different aspect ratios and scales. These boxes are evaluated based on their objectness score, which indicates how likely they contain an object. After the RPN generates region proposals, non-maximum suppression (NMS) is applied to remove redundant proposals. The final proposals generated by the RPN are fed into the Fast R-CNN detector, which classifies them and refines the bounding boxes. Importantly, RPN and Fast R-CNN share convolutional layers, meaning that both region proposal generation and object detection share computation. This makes Faster R-CNN extremely efficient. A visual representation of the RPN and Fast RCNN framework can be found in Figures 2 and 3 of the Appendix.

3.3.3 Fully Convolutional One-Stage Object Detection (FCOS)

FCOS (Fully Convolutional One-Stage Object Detector), as first proposed by Tian et al. [55], is an anchor-free object detection framework. This means that unlike the SSD and Faster R-CNN models, that rely on predefined bounding boxes, FCOS operates in a dense per-pixel manner by directly predicting bounding boxes for each location. By doing so, the model eliminates the need for predefined bounding boxes and region proposals. A visual representation of the FCOS framework can be found in Figure 4 of the Appendix.

As with many other object detection models, the FCOS framework uses a preexisting image classifier network as its model backbone (or base network). This backbone network processes the input image through several convolutional layers and generates the first feature maps. Using a Feature Pyramid Network (FPN), extra feature maps are generated at different sizes. Each feature map in the pyramid is responsible for detecting objects within a specific size range, allowing the FCOS model to handle objects of different shapes and sizes. These feature maps are then used as input for the prediction head. The prediction head processes each location of the feature map and generates outputs for object detection. The prediction head is mostly made up of convolutional layers and consists of three main branches: the classification branch, the regression branch, and the center-ness branch.

The classification branch outputs the probability of the object class of each pixel on the feature maps. It performs binary classification for each possible class, where a pixel is classified as either the background or a specific object class. The regression branch predicts the bounding boxes of objects. It treats each pixel in the feature maps as a possible object center and predicts four values: the distances from the current pixel to the left, right, top, and bottom of the bounding box. Finally, the center-ness branch predicts how close the pixel is to the center of the object. This score helps suppress low-quality detections that are generated by pixels far from the object center. The center-ness branch is a single convolutional layer that produces a single scalar output for each pixel in the feature maps. This output is used during inference to re-weight the classification score, helping to filter out low-confidence detections. It ranges between 0 and 1, with higher values indicating pixels closer to the center.

After the network outputs the predicted bounding boxes, class scores, and center-ness scores, FCOS applies NMS to filter out duplicate detections and select the most confident bounding boxes. The final score for each detection is computed by multiplying the classification score with the center-ness score, ensuring that detections from the center of the object are prioritised.

By avoiding anchor boxes and directly predicting bounding boxes at each location, FCOS achieves competitive detection accuracy with a much simpler architecture, making it a strong alternative for real-time object detection.

3.4 Related Work

While, at the time of writing, there have been no reported cases of the use of object detection models on X-ray images of hand luggage in real life yet, there has been research on the subject over the past couple of years. Most of this research can be divided into one of three categories: the comparison of multiple models for the same task, the proposal of a new model or security framework, or the use of synthetic data.

Literature like Petrozziello & Jordanov [41], Akcay et al. [2], Ackay & Breckon [1], and Mery et al. [33], all compare existing object detection models for the purpose of threat detection in X-ray images of luggage. More specifically, Petrozziello & Jordanov [41] explore the use of deep learning techniques to improve the detection of firearm components in X-ray scans of luggage. Two deep learning methods (CNNs and Stacked Autoencoders) are comapred with traditional classifiers such as Random Forrest, finding that CNNs consistently outperformed all other techniques across various metrics. A similar conclusion is reached by Akcay et al. [2], who focus on the application of deep learning techniques, particularly CNNs, for the detection and classification of objects within X-ray baggage imagery. In their research, CNN-based classification techniques also heavily outperformed traditional feature extraction methods. These studies demonstrate the significant potential of CNNs in the detection of dangerous objects in X-ray scans of luggage.
When comparing specific CNN frameworks, as done by Ackay & Breckon [1], it becomes apparent that region-based object detection methods such as Faster R-CNN show great potential. However, it is mentioned that future work should also compare other methods and advanced detection models like SSD within this context. Furthermore, the research shows that finetuning techniques can be used for training CNN models in this context when data or computational power is limited.

Besides comparing existing detection models and classifiers, new novel approaches to threat detection in luggage are also proposed. For example, in two separate papers Wang et al. [56] [57] explore the use of 3D Convolutional Neural Networks (CNNs) for the detection and classification of prohibited items within volumetric 3D CT baggage scans. The papers extends popular 2D object detection frameworks like Faster R-CNN and RetinaNet to work with 3D volumetric data, stating that this framework offers significant improvements over 2D methods.
Other literature like Wiley et al. [60] also propose the use of a different kind of scanning mechanism. Instead of the standard X-ray scanners, a CT scan is suggested for baggage scanning along with a novel approach to segment objects from CT scans of checked baggage for airport security. The method, called Stratovan Tumbler, was developed to handle challenges in CT imagery, such as noise, streaking artifacts, and complex object configurations. The key goal is to improve segmentation, which can assist in the automatic detection of explosives and other threats.

In the third and final subcategory, literature mainly relates to the use of synthetic data when creating or training models for the detection of prohibited items in baggage. Bhowmik et al. [7] investigate the application of CNNs for detecting prohibited items such as firearms and knives in X-ray baggage scans, with the primary focus on comparing the performance of CNN models trained on real versus synthetically generated X-ray images. The authors propose using synthetically composited images through a process called Threat Image Projection (TIP), where prohibited items are superimposed onto benign X-ray imagery. These synthetically created images aim to simulate the complexity and clutter seen in real baggage, thus enhancing the training process. Different CNN models are then evaluated on real, synthetic, and mixed (real + synthetic) datasets to observe differences in detection capabilities. Models trained purely on synthetic data performed worse when evaluated on real X-ray images, indicating that synthetic data alone cannot fully substitute real-world data. A combination of real and synthetic data did not significantly improve performance over using real data alone, suggesting that synthetically composited imagery may need further refinement. To improve performance, the paper suggests future work to improve the quality of synthetic imagery through Generative Adversarial Net-

works (GANs) to increase the variability and realism of prohibited items.
The use of GANs in this context is something as shown in literature like Kolte et al. [25], which focuses on the challenge of detecting prohibited objects in X-ray images using GAN-based anomaly detection methods. The paper addresses the issue of data scarcity for threat objects and presents a new ensemble method combining Skip-GANomaly and a modified UNet-style generator to improve performance. The study demonstrates that GAN-based models, particularly ensembles of GAN architectures, can be highly effective for threat object detection in X-ray imagery, and that the use of anomaly detection with GANs could help address the data scarcity problem.

As briefly mentioned in Chapter 1, there are two main problems with current literature. When looking at papers like Mery et al. [33], who provide a comprehensive overview of advancements in X-ray baggage inspection, it can be seen that almost all research is conducted using private or classified datasets. As a result, (numerical) findings cannot or only partially be reported and much of the literature becomes impossible to reproduce. This makes it difficult to make reliable comparisons between proposed approaches and hinders future research.
Furthermore, the new models and frameworks that are proposed often heavily rely on new or unusual scanning procedures. The research mentioned in this section (Wang et al. [56] [57], Wiley et al. [60]) is no exception, further indicating the problem. This means airports likely have to make major and expensive changes in their security infrastructure before they can adopt these new methods. As this is not only expensive but also risky, because these methods have no proven real world track record, the chance of adoption becomes small.

CHAPTER 4

Methodology

This chapter describes the general outline of the presented research. It provides a broad overview of the main research approach, justifying and explaining important choices. Furthermore, it will discuss the principles and methods needed when aiming to reproduce or build upon the research. Section 4.1 will give an overview of the different object detection approaches, followed by the object detection model selection. The training and evaluation of these models will be addressed in Sections 4.2 and 4.3. Finally, real world implication of the models will be taken into account and considered in Section 4.4.

4.1 Object Detection Model Selection

It is crucial that the correct models are chosen to produce relevant and insightful results. As mentioned in Section 3.4, there has been literature comparing traditional object detection models against more recent CNNs and other deep learning architectures for the specific task of object detection in X-ray scans of hand luggage. When looking at the results, it can be seen that CNNs always outperformed other models. That is why for this research, the choice is made to compare different CNN models for this specific task.

Since the adoption of CNNs in the realm of object detection, almost all CNN based object detection models can be divided into one of three groups: anchor-based single-stage, anchor-based two-stage, and anchor-free models. As each detection approach comes with its own

benefits and drawbacks, it is important to compare them for each task.
The detection of prohibited items in X-ray images is no exception. That is why, as also stated in Chapter 1, a large part of this research consists of comparing different CNN models for the specific task of detecting prohibited items in X-ray scans of hand luggage. This section will first compare the three main object detection approaches by highlighting their differences, advantages, and drawbacks. Furthermore, it will elaborate on the choices made during model selection and present the chosen object detection models.

4.1.1 Anchor-Based and Anchor-Free Approach

The first major distinction that can be made when comparing object detection models is whether the model is anchor-based or anchor-free.
As noted in literature like that of Zhang et al. [61], anchor-based models use predefined bounding boxes called ’anchors’ or ’priors’ that serve as a guide to predict the location and size of objects. Anchor-based models use a variety of predefined anchors, each with different shapes and sizes. During training the model tries to select the best fitting anchor for each object and refine it to match the ground truth bounding box as close as possible. In contrast, anchor-free models do not rely on predefined bounding boxes but instead try to predict key points of an input image, such as object centers or corners, which are then used to make the bounding box predictions. Instead of placing predefined anchors across the image grid, anchor-free methods make predictions at every location in the image (per pixel or per feature map position).

Because anchor-based models have many anchors at each location that need to be considered, many candidate regions need to be processed. Because of this, they often have a more complex model architecture and design compared to anchor-free models that avoid the complexity of anchor box generation, consideration, and refinement.
Due to their more complex architecture, anchor-based models often achieve a higher detection accuracy. However, as the predictions are largely based on predefined boundingbox sizes and shapes, anchor-based methods are not that flexible and may struggle with the detection of objects that have unusual shapes or sizes. Consequently, anchor-based methods are the most usefull

in scenarios where object sizes and aspect ratios are well-understood and do not show a lot of variance. In comparison, anchor-free models are more flexible and can handle objects of varying sizes and shapes more naturally. This often makes them applicable to a broader range of tasks and well suited in applications where object shapes, sizes, and aspect ratios vary greatly.

4.1.2 Single-Stage and Two-Stage Methods

Upon closer inspection of anchor-based methods, a second distinction can be made. As pointed out by literature such as Demetriou et al. [11] and Soviany & Ionescu [50], models following the anchor-based object detection approach can be divided into two main methods: single-stage or two-stage.
Single-stage methods try to simultaneously locate and classify objects in a single forward pass through the object detection network. As the name suggests, two-stage detection methods split this task into two parts. First, a network proposes regions of interest. These are parts of the input image where the network thinks objects are present. Subsequently, a second network takes the full input image along with the proposed regions of interest and makes predictions for each region separately.

Single-stage methods are usually less complex, as they skip the region proposal step, making them easier to deploy and faster to train. However, as two-stage methods are more complex, they typically have the ability to achieve a higher accuracy. Because of this, single-stage methods are best suited for applications requiring fast detection, while two-stage methods excel in detailed applications where even small detection errors can have significant consequences.

4.1.3 Selected Models

When considering the task of detecting contraband in X-ray scans of hand luggage, no single approach stands out as the obvious choice. Taking the main findings from Chapter 2 into consideration, the model needs to be both fast (to enhance efficiency) and perform well (to ensure safety). Additionally, the model needs to be flexible as it needs to be able to detect a variety of objects in different shapes and sizes. As each of the requirements correlates with the strength of a different approach (speed: anchor-based single-stage, accuracy: anchor-based two-stage, flexibility: anchor-free), a trade-off needs to be made. That is why during this research the choice is made to compare one model form each of the previously discussed object detection approaches.

As anchor-based single-stage model the Single Shot Multibox Detector, as described in Section 3.3.1, is chosen. Secondly, the Faster R-CNN network, as described in Section 3.3.2, is chosen as the anchor-based two-stage model. Finally, the choice is made to use the Fully Convolutional One-Stage Object Detection (FCOS) approach, described in Section 3.3.3, as the anchor-free model.
All selected models are state of the art and among the most popular object detection models in their respective group. Additionally, they serve as the basis for many object detectors that followed them, thus serving as a good representation for anchor-based single-stage, anchor-based two-stage, and anchor-free models.

4.2 Object Detection Training

With the adequate models selected, a training procedure needs to be determined for each model. To ensure the best possible results, the training procedures as described in the original papers of the selected models are used as inspiration. However, if performance is not hindered (or possibly improved), changes are implemented in the training procedure to make them as similar as possible and guarantee a fair comparison.

4.2.1 Single Shot Multibox Detector (SSD)

A crucial aspect of the SSD model is the generation of anchors for each feature map. Following the training procedure presented in the original paper by Liu et al. [30], the anchor boxes are designed so that specific feature maps learn to be responsive to particular scales of objects. Suppose m feature maps are used for prediction, the scale of anchors for each feature map is then computed using Equation 4.1. Here, s_{min} (0.2) is the scale of the lowest layer and s_{max} (0.9) the scale of the highest layer. All layers in between the lowest and highest layer are regularly spaced.

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m-1}(k-1), \quad k \in [1,m] \tag{4.1}$$

Besides different scales, the model also uses different aspect ratios to generate anchors, denoted by $\alpha_r \in \{1,2,3,\frac{1}{2},\frac{1}{3}\}$. Using these ratios, the width and height of each anchor can be calculated using Equations 4.2 and 4.3. For the aspect ratio of 1, a second anchor is added whose scale is determined by Equation 4.4. The center of each anchor is defined by Equation 4.5, where f_k is the size of the k-th square feature map. This results in a total of six anchors per feature map location.

$$w_k^r = s_k\sqrt{\alpha_r} \tag{4.2}$$

$$h_k^r = s_k/\sqrt{\alpha_r} \tag{4.3}$$

$$s_k' = \sqrt{s_k s_{k+1}} \tag{4.4}$$

$$(\frac{i+0.5}{f_k}, \frac{j+0.5}{f_k}), \quad i,j \in [0,f_k) \tag{4.5}$$

During training, the ground truth boxes are matched with the generated anchors. Each ground truth box is matched to the anchor box with the highest overlap. Additionally, any anchor box that overlaps a ground truth box with more than 50% is also matched. This allows the network to predict high scores for multiple overlapping anchors, instead of having to pick just one. Since most anchors do not get matched to any ground truth objects, there is an imbalance between matched and unmatched anchors. SSD addresses this by using hard negative mining, where only the hardest negative examples (those with the highest confidence loss) are used during training, other anchor boxes are discarded. The ratio of matched to unmatched boxes is kept at a maximum ratio of 3:1. For each matched anchor, the model predicts the offsets (difference in center coordinates, width, and height) between the anchor box and ground truth. These offsets are then used to make a final bounding box prediction.

Using this matching strategy, the training objective can be defined by the overall objective loss function as defined in Equation 4.6. As can be seen, the loss function is a weighted sum of the localisation loss (loc) and the confidence loss (conf), where N is the number of matched default boxes (to normalize the loss).

$$L(x,c,l,g) = \frac{1}{N}\left(L_{\text{conf}}(x,c) + L_{\text{loc}}(x,l,g)\right) \tag{4.6}$$

The confidence loss (L_{conf}) is a softmax loss over multiple classes and computes the cross-entropy between the predicted class scores and the ground truth class labels. This part of the loss function is used to measure how well the model classifies objects, and can be seen in Equation 4.7.

$$L_{conf}(x,c) = -\sum_{i\in \text{Pos}} x_{ij}^{p} \log(\hat{c}_i^p) - \sum_{i\in \text{Neg}} \log(\hat{c}_i^0) \tag{4.7}$$

Where:

- x_{ij}^p is an indicator for whether the anchor box i is matched to a ground truth box j of category p.
- $\hat{c}_i^p$ is the predicted probability of class p for the i-th default box.
- The sum over $i \in$ Pos represents the sum over all positive (matched) default boxes, and the sum over $i \in$ Neg is over all negative (unmatched) default boxes.
- $\hat{c}_i^0$ is the predicted probability of the background class for the i-th default box.

The localisation loss (L_{loc}) is a smooth$_{L_1}$ loss between the predicted anchor offsets and the ground truth anchor offsets. It is calculated only for the matched anchor boxes as unmatched boxes do not have a corresponding ground truth bounding box. This process is mathematically shown in Equations 4.8 and 4.9.

$$L_{loc}(x,l,g) = \sum_{i\in \text{Pos}} x_{ij} \text{smooth}_{L_1}\left(l_i - g_j\right) \tag{4.8}$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{4.9}$$

Where:

- x_{ij} is an indicator for whether the anchor box i is matched to a ground truth box j.
- l_i represents the predicted offsets (center coordinates $\hat{g}_x, \hat{g}_y$, width $\hat{g}_w$, height $\hat{g}_h$) for the i-th default box.
- g_j represents the ground truth box parameters for the matched box.

4.2.2 Faster R-CNN

As mentioned in Section 3.3.2, the Faster R-CNN model consists of two parts: a network that generates region proposals (RPN), and an object detection network (Fast R-CNN). Both networks have a separate training procedure and are later combined.
As stated by Ren et al. [43], the Region Proposal Network (RPN) is trained to predict regions of interest (RoIs) along with an associated "objectness" score, which indicates whether an object is present in a given region. To achieve this, the RPN uses predefined anchor boxes with a variety of scales and aspect ratios at each sliding window position on the feature map. In this case, anchor boxes essentially represent initial candidate regions that may contain objects. During training, the RPN then refines these anchors through bounding box regression to generate better RoI proposals.

The anchors are generated based on a combination of predefined scales and aspect ratios. Three scales of 128^2, 256^2, 512^2 pixels and three aspect ratios of 1:1, 1:2, 2:1 are used, leading to 9 anchors at each position. The center of each anchor box is aligned with the center of the sliding window position on the feature map. In other words, the sliding window determines where the center of the anchor box is placed. This means that all anchors at a certain sliding window position of the feature map share the same center.
During training, the RPN assigns a positive label to anchor boxes that have the highest overlap with a ground-truth object. In addition, anchor boxes that overlap more than 70% with any ground truth box are also labeled positive. Negative labels are assigned to anchors with an overlap of less than 30% with any ground truth box. Anchors with an overlap between 30% and 70% are ignored. This strategy ensures that the RPN is trained to recognise the best-matching

regions for object detection.

Using this matching strategy, the training objective of the RPN can be defined. The loss function, as shown in Equation 4.10, combines the classification and regression losses.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{\text{cls}}}\sum_i L_{\text{cls}}(p_i,p_i^*) + \lambda\frac{1}{N_{\text{reg}}}\sum_i p_i^* L_{\text{reg}}(t_i,t_i^*) \tag{4.10}$$

Where:

- i is the index of an anchor.
- p_i is the predicted probability of anchor i containing an object. p_i^* represents the ground-truth label and equals 1 if the anchor is positive, and 0 if the anchor is negative.
- t_i is a vector representing the predicted offsets of the anchor box (center x, cnter y, width, and height). t_i^* that of the ground-truth box associated with a positive anchor i.
- The term $p_i^* L_{reg}$ ensures that regression loss is only activated for positive anchors ($p_i^* = 1$) and is disabled otherwise ($p_i^* = 0$).
- The terms $N_{cls}(= 256)$ and $N_{reg}(= 2400)$ act as normalising constants, and $\lambda(= 1)$ as a balancing weight.

The classification loss (L_{cls}) indicates how well the model can predict whether an anchor box contains an object or belongs to the background (objectness score). It is defined as the log loss over two classes (object vs. no object) and can be seen in Equation 4.11. The regression loss (L_{reg}) helps adjust the predicted coordinates of the bounding boxes to better match the ground truth boxes. It uses the $smooth_{L1}$ loss between the predicted bounding box coordinates and the ground truth coordinates and can be seen in Equations 4.12 and 4.13.

$$L_{cls}(p_i,p_i^*) = -p_i^*\log(p_i) - (1-p_i^*)\log(1-p_i) \tag{4.11}$$

$$L_{reg}(t_i,t_i^*) = \text{smooth}_{L_1}(t_i - t_i^*) \tag{4.12}$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{4.13}$$

The object detection part of the model (Fast R-CNN network) has a different training procedure. As mentioned in Section 3.3.2, the Fast R-CNN model takes a full input image and a set of RoIs (generated by the RPN). Girshick [15] points out that the set of RoIs allows the model to adopt a single-stage anchor-free training strategy. As each training RoI is labeled with a ground-truth class u and a ground-truth bounding-box regression target v, the training objective can be defined by using a multi-task loss L on each labeled RoI to jointly train for classification and bounding-box regression. This multi-task loss is shown in Equation 4.14, where L_{cls} represents the classification loss and L_{loc} the localisation loss. To make sure that the localisation loss is only used in the loss function if an object is detected, $[u \geq 1]$ is 1 when $u \geq 1$ (and zero otherwise). The hyper-parameter λ controls the balance between the two task losses, during this research λ is set to one.

$$L(p,u,t^u,v) = L_{cls}(p,u) + \lambda[u \geq 1]L_{loc}(t^u,v) \tag{4.14}$$

During classification, the model outputs a discrete probability distribution over all class categories. To calculate the classification loss related to this output, the log loss is used as can be seen in Equation 4.15.

$$L_{cls}(p,u) = -\log(p_u) \tag{4.15}$$

Bounding box predictions are made by refining the given RoIs. For each RoI containing an object, the bounding box prediction layer outputs the predicted offsets needed to refine the RoI bounding box around the actual object. Using this output, the localisation loss is calculated by using Equations 4.16 and 4.17.

$$L_{loc}(t^u,v) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_u^i - v^i) \tag{4.16}$$

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{4.17}$$

Where:

- $t_u = (t_x, t_y, t_w, t_h)$ represents the predicted bounding box regression offsets for class u.
- $v = (v_x, v_y, v_w, v_h)$ represents the ground truth bounding box regression targets for class u.

In order to learn convolutional layers that are shared between the RPN and Fast R-CNN, a training technique needs to be adopted that allows for sharing convolutional layers between two networks. Note that simply defining a single network that includes both RPN and Fast R-CNN, and then optimizing it jointly with backpropagation is not a viable option. The reason is that Fast R-CNN training depends on fixed object proposals and it is not clear beforehand if the learning of Fast R-CNN while simultaneously changing the proposal mechanism will converge.
That is why, to train the entire system, an alternating training scheme is used as presented below:

1. Train the RPN separately.
2. Train Fast R-CNN separately using region proposals generated by the trained RPN.
3. Fine-tune the RPN with shared convolutional layers from Fast R-CNN.
4. Fine-tune Fast R-CNN while keeping the shared layers fixed.

This alternating optimisation ensures that both the RPN and Fast R-CNN networks learn shared features for region proposal generation and object detection.

4.2.3 Fully Convolutional One-Stage Object Detection (FCOS)

Following the training procedure of FCOS as presented by Tian et al. [55], locations (pixels) on the feature map are treated as either positive or negative samples. This distinction is based on their position relative to the ground-truth bounding boxes. Specifically, location (x, y) is considered a positive sample if it falls into the center area of any ground-truth bounding box. The center area of a bounding box centered at (c_x, c_y) is defined as the sub-box $(c_x - rs, c_y - rs, c_x + rs, c_y + rs)$, where s is the stride of the current feature map and $r(= 1.5)$ is a hyper-parameter determining the size of the sub-box. Positive samples get assigned a target label c^* which is the same as the class of the object in their corresponding ground truth bounding box. Besides a label for classification, positive samples are also assigned a regression target $t^* = (l^*, t^*, r^*, b^*)$. Here, l^*, t^*, r^*, b^* are the distances from the location to the four sides (left, top, right, bottom) of the bounding box. More specifically, if location (x, y) is associated to a ground truth bounding box B_i, where $(x_0^{(i)}, y_0^{(i)})$ and $(x_1^{(i)}, y_1^{(i)})$ denote the coordinates of the top left and bottom right corners of the bounding box. Then the training regression targets for the location are formulated as in Equation 4.18, where s denotes the current stride. If a location/pixel falls into the center area of multiple bounding boxes, it is considered as an ambiguous sample. For these locations, the bounding box with minimal area is chosen as its regression target. Negative samples are assigned the background class label and no regression target. These samples are only used to help the model distinguish the background from the objects of interest.

$$l^* = \frac{(x - x_0^{(i)})}{s}, \quad t^* = \frac{(y - y_0^{(i)})}{s}, \quad r^* = \frac{(x_1^{(i)} - x)}{s}, \quad b^* = \frac{(y_1^{(i)} - y)}{s} \tag{4.18}$$

Corresponding to the training targets, the final layer of the networks predicts a vector p for classification and a vector t for bounding box regression. Vector p contains the probabilities of the current location belonging to each class, vector t contains the four predicted distances to the bounding-box (l, t, r, b). Instead of training a multi-class classifier, a binary classifier is trained for each class separately.

Using the model outputs, the loss function is defined as shown in Equation 4.19. Here, N_{pos} is the number of positive samples, $\lambda(= 1)$ is a balancing factor, and $1_{\{c^*_{x,y}>0\}}$ ensures that the regression loss is only applied to positive samples.

$$L(\{p_{x,y}\}, \{t_{x,y}\}) = \frac{1}{N_{\text{pos}}} \sum_{x,y} L_{\text{cls}}(p_{x,y}, c^*_{x,y}) + \frac{\lambda}{N_{\text{pos}}} \sum_{x,y} 1_{\{c^*_{x,y}>0\}} L_{\text{reg}}(t_{x,y}, t^*_{x,y}) \tag{4.19}$$

As can be seen, the overall loss function is defined as a combination of classification loss (focal loss) and regression loss (Generalized IoU loss). The focal loss L_{cls} is applied to the predicted classification scores $p_{x,y}$ for each location (x,y) on the feature map. It penalizes confident predictions for easily classified background samples more heavily, focusing the model on harder samples. The focal loss is defined in Equation 4.20.

$$L_{\text{cls}}(p_{x,y},c^*_{x,y}) = \begin{cases} -\alpha(1-p_{x,y})^\gamma \log(p_{x,y}) & \text{if } c^*_{x,y} > 0 \\ -\alpha p^\gamma_{x,y} \log(1-p_{x,y}) & \text{if } c^*_{x,y} = 0 \end{cases} \tag{4.20}$$

where:

- $p_{x,y}$ is the predicted probability of the true class at location (x,y).
- $c^*_{x,y}$ is the ground-truth class label ($c^* = 0$ for background).
- $\alpha(= 0.25)$ is a balancing factor.
- $\gamma(= 2.0)$ is the focusing parameter.

The regression loss L_{reg} is computed only for positive samples (locations inside the center area of ground-truth boxes). FCOS uses the Generalized IoU (GIoU) loss, which improves over standard IoU loss by considering the shape and size mismatch between predicted and ground-truth boxes. The definition of the GIoU loss is given in Equation 4.21, where the GIoU is formulated as in Equation 4.22.

$$L_{\text{reg}}(t_{x,y},t^*_{x,y}) = 1 - \text{GIoU}(B_{\text{pred}},B_{\text{gt}}) \tag{4.21}$$

Where:

- B_{pred} is the predicted bounding box based on the regressed distances (l,t,r,b).
- B_{gt} is the ground-truth bounding box.
- $\text{GIoU}(B_{\text{pred}},B_{\text{gt}})$ is the Generalized IoU between the predicted and ground-truth bounding boxes.

$$\text{GIoU}(B_{\text{pred}},B_{\text{gt}}) = \frac{\text{area}(B_{pred} \cap B_{gt})}{\text{area}(B_{pred} \cup B_{gt})} - \frac{|C(B_{\text{pred}},B_{\text{gt}}) - B_{\text{pred}} \cup B_{\text{gt}}|}{|C(B_{\text{pred}},B_{\text{gt}})|} \tag{4.22}$$

Where:

- $B_{pred} \cap B_{gt}$ denotes the intersection of the predicted and ground truth bounding boxes.
- $B_{pred} \cup B_{gt}$ denotes the union of the predicted and ground truth bounding boxes.
- $C(B_{\text{pred}}, B_{\text{gt}})$ is the smallest enclosing box that can contain both B_{pred} and B_{gt}.
- $|\cdot|$ denotes the area of a box.

Using the above training scheme results in a number of low-quality detections produced by locations far away from the center of an object. To suppress these low-quality detections, a single layer branch that predicts the "center-ness" of a location is added in parallel with the regression branch. The center-ness is a value between 0 and 1, and depicts the normalized distance from the location to the center of the object that the location is matched to. Given the regression targets l^*, t^*, r^*, b^* for a location, the center-ness target is defined as in Equation 4.23. As the center-ness score is a value between 0 and 1, it is trained using the binary cross entropy (BCE) loss as stated in Equation 4.24. In this equation, $o_{x,y}$ is the predicted center-ness score and $o^*_{x,y}$ the ground truth. This loss is added to the loss function in Equation 4.19 to obtain the final overall loss function as stated in Equation 4.25. Note that as the the term $1_{\{c^*_{x,y}>0\}}$ is added before the center-ness loss to ensure that it only influences the loss of positive samples.

$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}} \tag{4.23}$$

$$L_{\text{cent}}(o_{x,y}, o^*_{x,y}) = -\left[o^*_{x,y}\log(o_{x,y}) + (1 - o^*_{x,y})\log(1 - o_{x,y})\right] \tag{4.24}$$

$$\begin{aligned} L(\{p_{x,y}\}, \{t_{x,y}\}, \{o_{x,y}\}) = & \frac{1}{N_{\text{pos}}} \sum_{x,y} L_{\text{cls}}(p_{x,y}, c^*_{x,y}) \\ & + \frac{\lambda}{N_{\text{pos}}} \sum_{x,y} 1_{\{c^*_{x,y}>0\}} L_{\text{reg}}(t_{x,y}, t^*_{x,y}) \\ & + \frac{1}{N_{\text{pos}}} \sum_{x,y} 1_{\{c^*_{x,y}>0\}} L_{\text{cent}}(o_{x,y}, o^*_{x,y}) \end{aligned} \tag{4.25}$$

4.2.4 Model Finetuning

As stated in Section 3.4, literature has shown finetuning to be a viable option when training object detection models on X-ray imagery of luggage. Not only did finetuning improve results in some cases, it also requires less data and computational power. Due to the fact that computational power was limited during this research, and the possible increase in performance, the choice has been made to use finetuning when training the selected object detection models.

As pointed out by Peng & Wang [40], model finetuning refres to the process of using a pre-trained model and adapting it to perform a new task. The pre-trained model is usually trained on a very large dataset and, after adaptation for a new task, trained further on a (smaller) task-specific dataset.

The main idea behind this approach is transfer learning, which means that the knowledge learnt by a model in one domain is transferred to another related domain. In the context of object detection this could mean that pre-trained models already have learned a large number of low-level and mid-level features (for example edge detection). This knowledge can be reused in other similar tasks, resulting in a lower training time, less required data, and possibly better performance.

4.3 Object Detection Evaluation

To evaluate the object detection models, the mean average precision (mAP) evaluation metric as described by Everingham et al. [12] is used. The mAP is by far the most widely used evaluation metric for object detection tasks and thus important to report in order to ensure a fair comparison to other literature and future research. The multi-step process to compute the mAP is described below.

The first step in calculating the mAP is defining true positives (TP), false positives (FP), and false negatives (FN). To determine this for object detection tasks, the intersection over union (IoU) is used. IoU calculates the fraction of overlap between two figures, an IoU value of 0.9 thus indicates that 90% of the two figures overlap. In the case of object detection these are the predicted bounding boxes and the ground truth bounding boxes. The formula for the IoU can be seen in Equation 4.26, where $B_p \cap B_{gt}$ denotes the intersection of the predicted and ground truth bounding boxes, and $B_p \cup B_{gt}$ their union.

$$IoU = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} \tag{4.26}$$

Using this value, it now becomes possible to define true positives, false positives, and false negatives for object detection tasks. If a predicted bounding box predicts the correct class of an object and correctly matches the ground truth bounding box with an IoU above 0.5 (minimum 50% overlap), it is labelled as a true positive. If a predicted bounding box does not match any ground truth bounding box with an IoU above 0.5 or predicts the wrong class, it is labelled as false positive. Finally, a ground truth bounding box that does not have any matching predicted bounding box with an IoU above 0.5 is labelled false negative.
Besides being the standard in literature, the threshold of 50% was set deliberately low to account for inaccuracies/subjectivity in bounding boxes of the ground truth data. For example, defining the bounding box for a highly nonconvex object is somewhat subjective. Furthermore, multiple detections of the same object in an image are considered false detections, meaning that 5 detections of a single object counts as 1 correct detection and 4 false detections.

Using these definitions, the precision and recall of the model can be calculated for each class. Precision is the fraction of correctly predicted objects (TP) out of all predicted objects (TP + FP). In other words, it measures how accurate the positive predictions of a model are. The corresponding formula can be found in Equation 4.27. The formula for the recall is given in Equation 4.28, this represents the fraction of correctly predicted objects (true positives) out of all actual objects (true positives + false negatives). To clarify, this metric gives an indication of how well the model is able to identify positive instances.

$$P = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \tag{4.27}$$

$$R = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{4.28}$$

As object detection models output multiple predictions with associated confidence scores for each class, it is possible to calculate the precision and recall at different confidence levels. At each threshold, the model may predict more or fewer objects, changing the counts of true positives, false positives, and false negatives. This results in different precision and recall values at different confidence thresholds. By using these precision and recall values, a Precision-Recall (P-R) curve is plotted for each class, where the x-axis represents recall and the y-axis represents precision.

The average precision (AP) for each class is defined as the area under its corresponding P-R curve and represents the performance of a model at detecting objects of a given class across a range of confidence thresholds. Following the method of Everingham et al. [12], the AP for each class is calculated using the formula given in Equation 4.29. After the AP is calculated for each class, the final mAP metric is computed by averaging the AP across all object classes. A formula for this final step can be seen in Equation 4.30, where C represents the number of classes and AP_i the average precision of class i.

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p(r) \tag{4.29}$$

$$mAP = \frac{1}{C} \sum_{i=1}^{C} AP_i \tag{4.30}$$

4.4 Practical Implication

Taking real-world implication into account, it can be noted that just using object detection models and their respective mAP measures is not enough when aiming to improve the situation as presented in Chapter 2. Although it is important that objects are recognised and located (as done by the object detection models), the ability to differentiate between negative and positive samples has way more influence on the actual security process. The reason for this is that the distinction between positive and negative samples determines which bags can immediately be sent through, and which become subject to additional screening. That is why this will determine the actual improvement of passenger flow, not the detection of a specific object within the positive samples. Taking this into consideration, a second evaluation metric is added: background detection.

4.4.1 Background Detection

The background detection evaluation metric measures the performance of a model in the binary classification of positive and negative X-ray scans of hand luggage. Because there is a very large class imbalance (way more negative samples compared to positive samples), the accuracy will not give a good indication of model performance. This becomes clear when considering a model that always predicts the negative class. This model does not perform well, however, would still achieve a high accuracy score. To solve this, the F1-score is used as presented in Equation 4.31, where precision is calculated as stated in Equation 4.27 and recall as in Equation 4.28.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.31}$$

As can be seen, the F1-score combines the precision and recall into a single metric. By doing this, a model needs to achieve both a high precision and recall in order to perform well. Because of this, the F1-score gives a better indication of model performance for this task.
Using the background detection metric, three approaches are compared: using the original object detection models, extending the object detection models, adding a classification model.

The first considered approach is to use the object detection models as described in Section 4.1. Using the models, only trained on positive data, but testing them on both positive and negative samples allows the calculation of the background detection as described below.
If the object detection model does not detect any prohibited items with confidence greater than a threshold of 0.7, it predicts the negative class. If the model does detect items with enough confidence, it predicts the positive class. Using this method. true positives, true negatives, false positives, and false negatives can be defined. A true positive/negative is counted if the model predicts the positive/negative class and matches the ground truth. A false positive/negative is counted if the prediction is positive/negative and does not match the ground truth.

A second option is to extend the object detection models by adding a background class to the class options. Doing this allows the models to be trained on both positive and negative samples, stating that the background class should be predicted for negative samples. The main difference when compared to the first option is that the model now actively tries to detect negative samples instead of predicting the negative class if it cannot find any other items.
If the background class is detected with higher confidence than any other detection, the sample will be classified as negative, the detection of any other classes with higher confidence results in a prediction of the positive class. True/false positives and negatives are defined the same as in the previous approach.

The third and final considered approach is to use separate models for object detection and binary classification. By using two models, a pipeline is created where X-ray images of baggage are first processed by a binary classifier. Negative samples can leave the system (no prohibited items and nothing to detect), positive samples are sent to an object detection model that does the further classification and detection. In this case, the F1-score of the classification model is equal to the background detection metric. A visual representation of this process can be found in Figure 4.1.

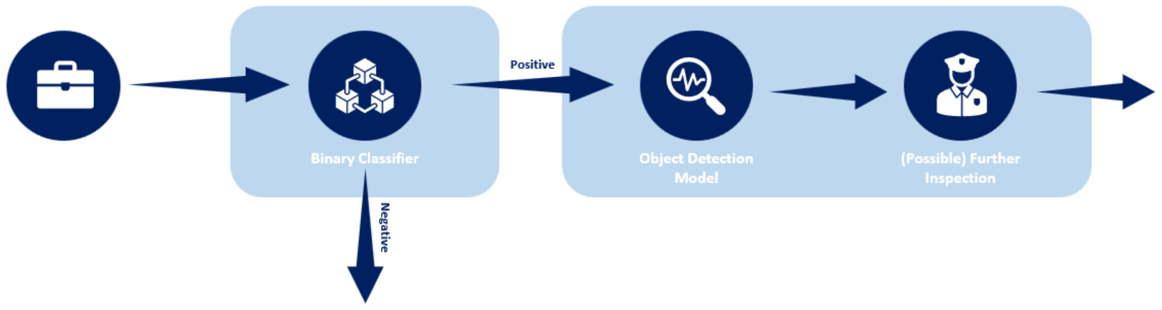

Figure 4.1: Background detection model pipeline.

An interesting trade-of that is applicable to all approaches is the ratio of FP and FN. Choosing to minimize the false negatives will lead to the safest possible model, as the least amount of positive samples will go unnoticed. However, this will probably come at the cost of an increased number of false positives, thus decreasing efficiency as more luggage will unnecessarily be selected for additional screening. The exact opposite happens when choosing to minimize false positives in order to optimize efficiency. This will in turn likely increase the false negatives, leading to a decrease in safety.

The same holds for the precision and recall metrics. As mentioned before, precision shows how accurate the positive predictions are, while recall states how well the model is able to detect positive cases. In this situation high precision thus indicates an efficient model, as a low rate of luggage gets unnecessarily checked. On the other hand, high recall indicates a very safe model, as almost no positive instances get missed.

This trade-off can be influenced by setting different confidence thresholds when determining FP, FN, TP, and TN and needs to be taken into account when implementing a final model.

4.4.2 Binary Classification Model Selection

When considering potential classification models for the model pipeline (approach 3), two main factors are taken into account: the model needs to perform well, and the model needs to be cheap to run. As discussed numerous times during this research, it is crucial that models perform well in order to ensure the safety of passengers. However, as Section 2.3 shows, security costs also need to be taken into account.

Because almost all samples are negative, the classification model in the described pipeline will do the majority of the computing. Making this model cheap to run, while upholding performance, can thus have a large influence on the computational costs of the system as a whole. That is why mainly image classifiers that are especially designed with efficiency in mind are considered. Table 4.1 shows the selection of models that are considered during this research. As can be seen, all models have a relatively small number of parameters of around 5-10 million (compared to the usual 50-100+ million of other state of the art image classifiers) while still achieving good performance in relevant literature. The models that are chosen are smaller variants of the ResNet, MNASNet, EfficientNet, and RegNet models as described in Sections 3.2.1, 3.2.2, 3.2.3, and 3.2.4 respectively.

Model	Number OF Parameters	Accuracy on ImageNet-1K
ResNet18	11.2M	89.078
EfficientNet B3	10.7M	96.054
MNASNet 1_3	5.0M	93.522
RegNet X_1_6GF_V2	8.3M	94.922

Table 4.1: Model Comparison: Number of Parameters and Accuracy

4.4.3 Binary Classification Training

Just as with the object detection models, a training procedure needs to be defined for the selected classification models. As the task of binary classification is overall simpler (compared to object detection) and the selected classification models are all similar in in/output, the same training procedure can be adopted for all selected image classifiers.
First, the large class imbalance needs to be addressed. In order to avoid problems that often arise when classifying heavily imbalanced datasets (like bias towards the majority class or poor generalisebility), undersampling will be used during training. More specifically, the majority class (negative samples) will be undersampled when creating the train set in order to guarantee a balanced distribution (of about 1:1) of positive and negative samples during training.

All selected models output their prediction in the same format. The models output a vector of raw unnormalised scores, known as logits. In the case of binary classification, this vector has a length of two: one logit for the positive class, and one for the negative class. Using the Sigmoid function, the logits are transformed into probabilities. This makes it possible to use the Binary Cross Entropy (BCE) Loss to train all models. The loss function is given in Equation 4.32.

$$L_{\text{BCE}}(p, c^*) = -\left[c^* \log(p) + (1 - c^*) \log(1 - p)\right] \tag{4.32}$$

Where:

- p is the predicted probability for the positive class,
- c^* is the ground-truth label, $c^* = 1$ for the positive class and $c^* = 0$ for the negative class.

CHAPTER 5

Experimental Setup

This Chapter will provide the full experimental setup of the research as described in Chapter 4, providing all the details needed when aiming to reproduce or build upon the given research. To ensure correct data selection and a full understanding of said data, Section 5.1 will discuss the data collection process, the used dataset, data preprocessing, and data augmentation. Next, this Chapter will give a detailed description of the training and evaluation setup of all used models in Section 5.2.

5.1 Data

5.1.1 Data Collection

Before any models can be trained, data needs to be collected. Collecting the right data is of the utmost importance to the success of a research, that is why there are important requirements that should be taken into account when searching for appropriate data. As stated in Chapters 1 and 3, most related literature use classified datasets. This makes it challenging and sometimes even impossible to compare or reproduce results. In order to solve this problem, the dataset used for this research should be open-source, making the results completely transparent and easily reproducible. Furthermore, the models should be trained on a dataset that gives an appropriate representation of real-world scenarios. The data should therefore reflect the current X-ray

screening output at airports as closely as possible. This will ensure reliable results and make sure trustworthy conclusions can be taken from the research.
To guarantee proper data collection, the current most commonly used X-ray screening technology must first be understood. In their paper, Graves et al. [18] provide a brief overview of this process.

Currently available X-ray screening technologies involve exposing the object of interest to X-ray radiation. When X-ray photons encounter solid matter, one of three things can happen: they can be absorbed, transmitted through the material, or scattered. The likelihood of each outcome is influenced by the energy of the X-ray radiation and the bulk properties of the material being examined. Examples of such properties include density, mass absorption coefficient, and effective atomic number (Zeff).
Zeff is a parameter used to describe how a certain material interacts with radiation, particularly X-rays or gamma rays. It is an average atomic number that accounts for the contributions of all the elements in the material. Due to the fact that many objects that are X-rayed do not consist of a single element but rather a combination of elements (plastics, organic materials, or alloys), Zeff is used to characterise the material's behaviour when exposed to ionising radiation (like X-rays). In the context of X-ray screening, knowing the Zeff allows operators and computer algorithms to distinguish between different materials. This distinction is important for detecting threats or prohibited items in security screening environments. For instance, materials with a low Zeff, typically organic substances, produce significant amounts of backscatter radiation, whereas materials with a higher atomic number, such as metals, tend to absorb most of the X-rays and exhibit minimal backscatter.

During the X-ray scanning process, the extent to which X-rays are transmitted or backscattered from objects is recorded. This data is processed and then used to generate an output image for interpretation. The visual representation of this information varies depending on the type of X-ray technology. Images can be presented in grayscale, where denser materials appear darker than lighter, organic elements. However, more often artificial colours are assigned to different materials based on their measured Zeff values. For example, low-Zeff materials are typically coloured orange, while high-Zeff materials appear green. Furthermore, properties such as mass density, nitrogen content, and Zeff values can be calculated for each material to facilitate accurate identification during the screening process.

Taking the above technical and practical requirements into account, two possible open-source datasets are considered: the SIXRay dataset as presented by Miao et al. [36], and the GDXray dataset presented by Mery et al. [34]. Both contain images of X-ray scans collected as described above, however, there are some major differences. First of all, the SIXRay dataset contains way more datapoints (about 1.1 milion v.s. 8000). Secondly, the SIXRay dataset provides real-world coloured images of both positive and negative samples, whereas GDXray only contains black and white positive samples that are collected in a closed environtment. This means that the SIXRay dataset not only provides a lot more data, but also contains more realistic properties compared to GDXray. Because of this, the SIXRay dataset has been chosen to use during this research, a more detailed description of the dataset will be given below in Section 5.1.2.

5.1.2 SIX-Ray Dataset

As stated in Section 5.1.1, the dataset used during this research is the Security Inspection X-ray (SIXray) dataset presented by Miao et al. [36]. As mentioned in their paper, this open source dataset consists of 1,059,231 X-ray images of hand luggage. The images were collected from several subway stations with the original meta-data indicating the presence or absence of prohibited items. The prohibited items are divided into six categories: gun, knife, wrench, pliers, scissors, and hammer (Table 5.1). If one of these items is present in an image it is labelled as positive, if this is not the case it is labelled as negative. Furthermore, in the case of a positive instance, the location of the prohibited item was manually annotated. The annotation for each positive instance is given in separate .xml files. Among other details such as object height and width, these files contain the class name and location of prohibited objects in the correlating image. The location of objects is given by stating the minimum and maximum values of the X and Y coordinates of the bounding boxes. Multiple prohibited objects of different classes, shapes, and sizes can be present in a single positive instance. Lastly, each image is obtained from a security X-ray scan, which assigned different colours to objects made of different materials, and is stored in JPEG format. The average size of each image is around 100K pixels.

Four main properties of this dataset make it a realistic representation of the presented research problem. Firstly, the presented data set raises a brand new challenge of overlapping image data. As mentioned above, the images were obtained from X-ray scans of personal luggage, in which objects are often randomly stacked. When these items are passed through an X-ray scan, the penetration property makes it possible to see even the occluded items in the image. This leads to the most important property of this dataset, which the authors call overlapping. This property is not present in any other open source dataset on this topic. Secondly, prohibited items can appear on many different scales, viewpoints, styles, and even subtypes. This causes considerable intra-class variation and increases the difficulty of recognition, just as in real life. The third property is the complex and meaningless contexts present in each image. Images can be heavily cluttered, making it sometimes difficult to assign clear class labels to objects. Finally, a very large class imbalance. SIXray contains more than one million X-ray images in which less than 1% of the instances have positive labels (i.e. prohibited items are annotated).
In conclusion, the SIXRay dataset adequately mimics a testing environment similar to real-world airport scenarios where inspectors often see greatly varying overlapping prohibited items appearing with a very low frequency. To give a clearer idea of the used data, examples of both positive and negative samples can be found in Figures 5-11 of the Appendix.

SIXRay Dataset (1,509,231)						
Positive (8,929)						**Negative**
Gun	Knife	Wrench	Pliers	Scissors	Hammer	
3,131	1,943	2,199	3,961	983	60	1,050,302

Table 5.1: SIXRay dataset class distribution

5.1.3 Data Prepocessing

Looking at Table 5.1, it can be seen that the 'Hammer' class only contains 60 samples. Because this is not nearly enough data for models to learn during training, the choice has been made to delete these samples from the dataset. In addition, the data has to be further preprocesed for both object detection and image classification to be used during training and evaluation.

In the case of object detection, five main operations are performed to process the data: data selection and splitting, parsing annotations, creating JSON input files, creating custom datasets, and initialising the data-loaders.
First a method is written that randomly samples image IDs from the SIXRay dataset. This can be done in two ways: randomly sampling IDs of positive instances only, or randomly sampling both negative and positive sample IDs at a given ratio. Using this method, sets of training, validation, and test IDs can be sampled. To make sure there is no overlap, the ID of an instance is removed from the sampling pool after being sampled. The train, val, and test subsets of sampled image IDs are saved in separate text files to be used in a later stage of the data preprocessing. Secondly, a method is created that parses the annotation of positive instances. Using the ElementTree library, the annotation file associated with a given image ID is loaded. From this file, all object labels of prohibited items in the image along with the minimum/maximum X and Y values of their corresponding bounding box are extracted. These values are saved in a dictionary containing the boundingbox coordinates and class labels of the given input image.

Using the two described methods, the JSON input files can be created. For each data subset (train, val, test), two JSON files are created. One containing the path to each sampled input image, and one containing all corresponding class labels and boundingbox coordinates. By looping over the earlier mentioned text files containing the sampled image IDs, the path to each image is generated and added to a JSON file. In the same loop, image IDs are passed to the described parsing method and the returned dictionary containing the class labels and boundingbox coordinates is added to another JSON file. If a negative instance needs to be parsed (that does not have an associated annotation file), a dictionary containing the background class

label and min/max X and Y values of the image is added to the JSON file. This results in a total of six JSON files, two for each subset (train, val, test), where one contains the path to the input images and the other the corresponding target classes and boundingbox coordinates.
Subsequently, the JSON files are used to create custom Pytorch datasets for the train, val, and test set. The datasets load the corresponding JSON files and implement a 'get_item' function. This function uses the provided image path to open the corresponding input image, convert it to RGB, and store it in a tensor. Simultaneously, the dictionary containing class labels and boundingbox coordinates of the image is extracted from the second JSON file and transformed in a dictionary of tensors. The method returns both the the image tensor and target dictionary (containing a tensor of class lables and a tensor of boundingbox coordinates).
Finally, the datasets are initialised, normalised, and passed to Pytorch dataloaders, resulting in a train, val, and test dataloader containing the input and target tensors. These dataloaders will be used to train and evaluate the object detection models.

Because binary classification models only need an input image and label (positive or negative), the data preprocessing is a lot simpler. By using the same sampling method as described above, both positive and negative image IDs are randomly sampled at a given ratio. Using the sampled image IDs, new folders are created for the train, val, and test set containing the actual images. These folders contain two subfolders each, one with positive sampled instances and one with negative sampled instances. The folders are passed to a Torchvision ImageFolder, which turns both the input images and lables into tensors, normalises the images, and stores them in the Python environment. At last, the ImageFolders are used to create three dataloaders, one for the training set, one for the validation set, and one for the test set.

5.1.4 Data Augmentation

Literature like Kaur et al. [20] and Zoph et al. [64] points out that data augmentation can improve the performance of object detection models. When looking at the original research presenting the used object detection models (liu et al. [30], Ren et al. [43], Tian et al. [55]), it can be seen that data augmentation is used as well. The Faster R-CNN model uses image flipping during training and the SSD model extends this by also implementing cropping and expanding. Using these operations as inspiration, data augmentation was implemented for this research on the training data. Five transformation methods are used to augment the data: flipping horizontally, flipping vertically, randomly rotating, expanding, and cropping.

Horizontal and vertical flipping involves the process of flipping the input image along with the bounding boxes. When rotating, the image is randomly rotated between -180 and 180 degrees around the center. The expand transformation performs a zooming-out operation by randomly placing the original image in a larger canvas of filler material. Besides making the model more robust, this also has potential to increase the detection of smaller objects. Lastly, a crop operation can be performed. This transformation randomly crops the image around an object. First, a number k is sampled from the list [0.1, 0.3, 0.7, 0.9]. A random patch of the image is then sampled and it is checked if the random patch contains at least k% of a boundingbox along with the object center. If this is the case, the new image is returned, otherwise a new patch is sampled and checked again. After a correct patch is found, boundingboxes are updated and the transformed image is returned. Instead of helping the model detect smaller objects, this transformation helps the model detect larger and partial objects.
Flipping/rotating operations are applied with a probability of 0.5, the image is then either expanded or cropped with a probability of 0.25. All original input images are always used during training, the transformed images act as an extension of the dataset and do not replace any original images. The transformation of a sample is visualised in Figures 13 - 17 of the Appendix.

5.2 Model Training and Evaluation

Following the training techniques presented in Sections 4.2 and 4.4.3, the object detection and classification models are trained for multiple scenarios. To ensure a fair comparison, all detection models are trained using the Adam optimiser with a learning rate of 0.0001 and a weight decay of 0.0005. Furthermore, all models are trained using the Google Colab T4 GPU in the Google Colab environment.

5.2.1 Training and Evaluating Object Detection Models

First, the Torchvision package is used to load the SSD, Faster R-CNN, and FCOS model, using weights pretrained on the COCO dataset. Two changes are made to the standard models as provided by Torchvision: resizing the output layers, and changing the backbone network. To ensure that the models are able to detect the correct amount of classes (5 in total: knife, gun, wrench, pliers, scissors), the output layers are resized. For the SSD and FCOS model, this is achieved by resizing the regression and classification heads. For the Faster R-CNN model, the box predictor layer is resized. Secondly, if not already the case, the backbone network of the model is changed to the ResNet50 network (also with weights pretrained on the COCO dataset), ensuring a fair comparison of the models.

Next, three sets of dataloaders are made by following the process as explained in Section 5.1.3. The structure of the data loaders is shown below:

Set 1:

- Training dataloader containing image/target tensors of 80% of positive samples (randomly sampled).
- Validation dataloader containing image/target tensors of 10% of positive samples (randomly sampled, no overlap).
- Test dataloader containing image/target tensors of 10% of positive samples (randomly sampled, no overlap).

Set 2:

- Training dataloader containing image/target tensors of 80% of positive samples (randomly sampled).
- Validation dataloader containing image/target tensors of 10% of positive samples (randomly sampled, no overlap).

- Test dataloader containing image/target tensors of 10% of positive samples and image/target tensors of 100x more negative samples (randomly sampled, no overlap). Resulting in the real-world ratio of pos:neg = 1:100.

Set 3:

- Training dataloader containing image/target tensors of 80% of positive samples and image/target tensors of negative samples at the same rate as the amount of samples containing the hammer class (randomly sampled, no overlap). By doing this, the background class is distributed roughly the same in the training set as the other object classes, allowing balanced training.
- Validation dataloader containing image/target tensors of 10% of positive samples and image/target tensors of 100x more negative samples (randomly sampled, no overlap). Resulting in the real-world ratio of pos:neg = 1:100.
- Test dataloader containing image/target tensors of 10% of positive samples and image/target tensors of 100x more negative samples (randomly sampled, no overlap). Resulting in the real-world ratio of pos:neg = 1:100.

Using set 1 of the dataloaders, all object detection models are first trained on positive data. During training, a maximum of 10 epochs and a batch size of 4 is used. Furthermore, training is stopped early if validation loss starts to increase, as this indicates overfitting. The choice for a small batch size of four is made due to computational limitations, as larger batch sizes took up to much memory space of the T4 GPU.

As mentioned in Section 4.2.4, model finetuning is used during training, Table 5.2 shows the parts of the object detection models that were frozen. First, only the last parts of the models (SSD classification head, RoI head, FCOS head) are trained, the other parts are frozen. The trained models are validated on the validation set using the mAP metric as described in Section 4.3. After this, the models are reinitialised and trained/validated again. This time, both the first and second parts of the models are unfrozen, meaning that only the backbone network is left frozen for all models. Results of this process can be found in Table 6.1.

SSD	Faster R-CNN	FCOS
Backbone (ResNet)	Backbone (ResNet)	Backbone (ResNet)
SSD regression head	RPN	Feature pyramid network
SSD classification head	RoI heads	FCOS head

Table 5.2: Parts of the object detection models that are frozen during finetuning.

The best performing SSD, Faster R-CNN, and FCOS models are saved and finally evaluated on the test set. To ensure trustworthy results, this process is repeated four more times by using different seeds when sampling data for the dataloaders, results can be found in Table 6.2.
To further inspect the effect of freezing different layers, the models are trained again following the exact same training setup as described above. However, this time without freezing any layers. Training all layers produced the results as seen in Table 6.3. Furthermore, to try and improve results of the best performing models, data augmentation from Section 5.1.4 was implemented. All other parts of training are kept the same. The results of implementing data augmentation can be found in Table 6.4.

Next, the background detection metric is also taken into account. The same training setup as described above is followed but this time using the second set of dataloaders. As testing is now done on both positive and negative instances, it becomes possible to calculate the background detection metric as described in Section 4.4.1. It is important to note that currently the used models are still only trained on positive samples, the technique of training on both positive and negative samples is explored later. To ensure reliable results, even with the large class imbalance of the test set, this process is repeated multiple times using 5-fold cross-validation. Results are presented in Table 6.5.
In an attempt to improve background detection while maintaining mAP levels, the object detection models are extended as explained in Section 4.4.1. This is done by again resizing the output layers of the models as done earlier, but now adding one more class for the background. As explained before, this allows the model to be trained on both positive and negative data. Using the generated train-, val-, and testloaders from set 3, the models are retrained and evaluated. Again, 5-fold cross-validation is used, the results of which are presented in Table 6.6.

5.2.2 Training and Evaluating Binary Classification Models

Following the last part of Section 5.1.3, dataloaders are created for training the image classification models. The structure of the dataloaders is presented below.

- Training dataloader containing image/target tensors of 80% of positive samples and image/target tensors of negative samples at the same ratio (randomly sampled). Resulting in the balanced ratio of pos:neg = 1:1.
- Validation dataloader containing image/target tensors of 10% of positive samples and image/target tensors of 100x more negative samples (randomly sampled, no overlap). Resulting in the real-world ratio of pos:neg = 1:100.
- Test dataloader containing image/target tensors of 10% of positive samples and image/target tensors of 100x more negative samples (randomly sampled, no overlap). Resulting in the real-world ratio of pos:neg = 1:100.

After creating the train, test, and val set, the classification models are loaded with random weights using the Torchvision package. Just as with the object detection models, the output layers are resized, this time for binary classification.
Using the created training, validation, and test set, the models are all trained following the training approach presented in Section 4.4.3. The models are trained for a maximum of 10 epochs, stopping early if validation loss started to increase. Furthermore, all models are trained using a batch size of 32. After training, the models are evaluated on the test set using the recall, precision, and F1 evaluation metric. Once again, to make sure results are credible even with the high class imbalance in the test set, 5-fold cross validation is used when training the image classifiers. The results of this process are shown in Table 6.7.

CHAPTER 6

Results

This Chapter contains the results of the presented research. It shows the results obtained following the experiments described in Chapters 4 and 5. First, the results related to the object detection task are presented in Section 6.1. Afterwards, the scope is extended to also take the previously mentioned background detection task into account, these results are addressed in Section 6.2.

6.1 Object Detection

Table 6.1, shows the results of freezing different parts of the selected object detection models when trained on positive data. As can be seen, freezing fewer layers results in better performance of all models on the validation set. Looking at these initial results, the Faster R-CNN model shows the best performance with the highest mAP value of 0.698. The FCOS model obtains the second highest mAP score with a value of 0.634. Finally, the SSD model performs the worst, achieving the lowest mAP of 0.252.

SSD		Faster R-CNN		FCOS	
Frozen Parts	**mAP**	**Frozen Parts**	**mAP**	**Frozen Parts**	**mAP**
Backbone + regression	0.154	Backbone + RPN	0.539	Backbone + FPN	0.473
Backbone	0.245	Backbone	0.718	Backbone	0.647

Table 6.1: Validation results of object detection models while freezing different parts.

Table 6.2 presents a more detailed view of how the models perform on the test set when the backbone is frozen. Looking at the results, it can be seen that the SSD model still gets outperformed by the other models, achieving the lowest average mAP score of 0.231. Faster-RCNN performs the best compared to the other models with an average mAP score of 0.721, closely followed by the FCOS model with an average mAP of 0.662.

SSD		**Faster R-CNN**		**FCOS**	
Class	**AP (σ)**	**Class**	**AP (σ)**	**Class**	**AP (σ)**
Knife	0.301 (0.094)	Knife	0.774 (0.051)	Knife	0.634 (0.059)
Gun	0.600 (0.048)	Gun	0.915 (0.021)	Gun	0.851 (0.028)
Wrench	0.071 (0.124)	Wrench	0.611 (0.059)	Wrench	0.502 (0.055)
Pliers	0.109 (0.090)	Pliers	0.635 (0.061)	Pliers	0.596 (0.057)
Scissors	0.070 (0.155)	Scissors	0.672 (0.061)	Scissors	0.729 (0.045)
mAP	0.231 (0.098)	mAP	0.721 (0.033)	mAP	0.662 (0.029)

Table 6.2: Results of object detection models using finetuning (freeze backbone).

Not freezing any layers produced the results in Table 6.3. An increase in mAP can be seen for all models. Although the SSD model makes the greatest improvement, it still achieves the lowest average mAP (0.721). The FCOS model shows the second best performance with an average mAP of 0.843 but is again outperformed by the Faster R-CNN model (average mAP of 0.873). Looking at the average AP of all classes separately, it can be seen that the Faster R-CNN model shows the highest average AP in the detection of all classes except scissors, which are best detected by the FCOS model. It can also be seen that all models obtain the highest AP when detecting the gun class. Furthermore, anchor-based models (SSD and Faster R-CNN) have the most difficulty when detecting scissors, whereas the anchor-free FCOS model obtains the lowest average AP for the detection of the wrench class.

To try and improve performance even more, data augmentation as introduced in Section 5.1.4 is implemented, resulting in Table 6.4. Results show that data augmentation has a negative effect on the performance of all models. Upon closer inspection, this negative effect is found to be the result of the cropping operation. All other transformations have no significant effect on model performance. Because of this, transformations are omitted from all further model training.

SSD		Faster R-CNN		FCOS	
Class	**AP (σ)**	**Class**	**AP (σ)**	**Class**	**AP (σ)**
Knife	0.738 (0.033)	Knife	0.863 (0.025)	Knife	0.782 (0.037)
Gun	0.928 (0.014)	Gun	0.947 (0.009)	Gun	0.905 (0.23)
Wrench	0.711 (0.054)	Wrench	0.851 (0.049)	Wrench	0.781 (0.042)
Pliers	0.777 (0.049)	Pliers	0.893 (0.036)	Pliers	0.891 (0.028)
Scissors	0.706 (0.046)	Scissors	0.813 (0.039)	Scissors	0.854 (0.025)
mAP	0.772 (0.021)	mAP	0.873 (0.019)	mAP	0.843 (0.022)

Table 6.3: Results of object detection models after training all layers (no freezing).

	SSD	Faster R-CNN	FCOS
mAP without data augmentation	0.772	0.873	0.848
mAP with data augmentation (all transformations)	0.704	0.796	0.751
mAP with data augmentation (no cropping)	0.766	0.869	0.851

Table 6.4: The effect of data augmentation on model performance.

Figure 6.1 shows the training graphs of all models. As can be seen, all models trained for the full 10 epochs without overfitting and converged nicely during each experiment. The Faster R-CNN model took the longest to train with an average training time of about 6 hours, followed by FCOS (4 hours) and finally by the SSD (1.5 hours). Inference time for all models is less than one second.

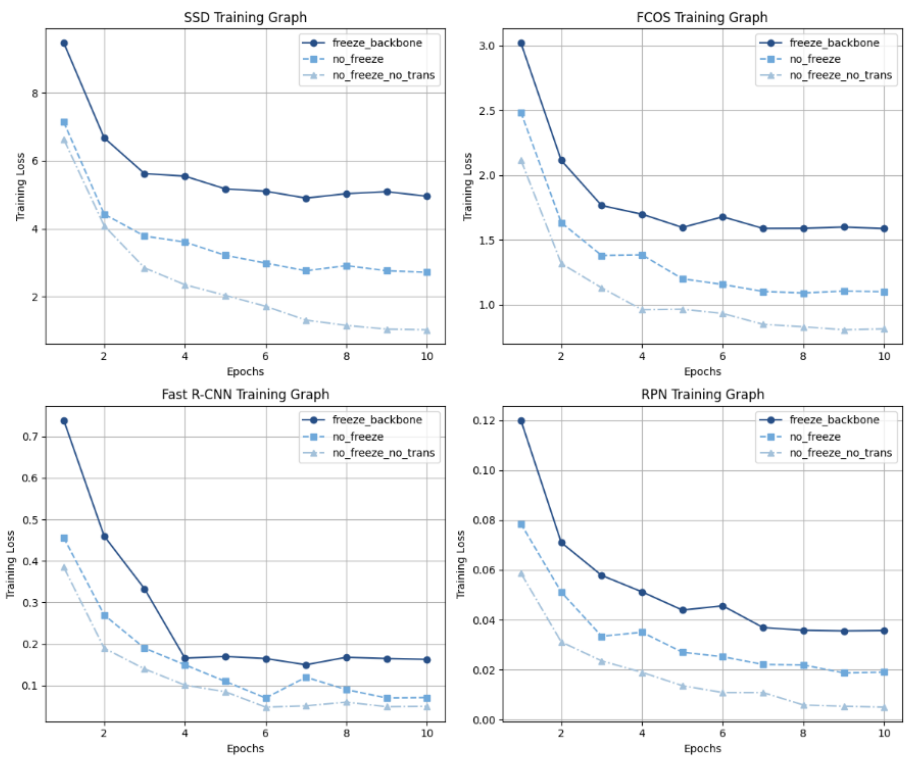

Figure 6.1: Training graphs of object detection models.

6.2 Background Detection

Next, the object detection models are evaluated in a wider scope. Taking Table 6.5 into account, it can be seen that all original object detection models (training all layers on positive data) continue to perform well, even when evaluated on both positive and negative data. As expected, there is no significant change in mAP compared to the results of Table 6.3. Looking at the background detection metric (denoted in the table as F1), it can be seen that the Faster R-CNN model achieves the best result with a F1-score of 0.852, once again outperforming both the SSD and FCOS models with a F1-score of 0.773 and 0.816 respectively.

As explained in Sections 4.4.1 and 5.1.3, the original object detection models are extended in an attempt to improve background detection (while maintaining mAP). The results of the extended models are shown in Table 6.6. Looking at the results, it can be seen that extending the object detection models does not increase their performance in any way. More specifically, all models show a decrease in their ability of both object and background detection.

SSD		Faster R-CNN		FCOS	
CLASS	**AP (σ)**	**CLASS**	**AP (σ)**	**CLASS**	**AP (σ)**
Knife	0.726 (0.038)	Knife	0.854 (0.033)	Knife	0.776 (0.045)
Gun	0.903 (0.019)	Gun	0.948 (0.025)	Gun	0.897 (0.021)
Wrench	0.697 (0.055)	Wrench	0.833 (0.048)	Wrench	0.752 (0.047)
Pliers	0.778 (0.029)	Pliers	0.852 (0.016)	Pliers	0.901 (0.013)
Scissors	0.715 (0.035)	Scissors	0.827 (0.019)	Scissors	0.850 (0.015)
mAP	0.764 (0.019)	mAP	0.863 (0.023)	mAP	0.835 (0.017)
F1	0.773 (0.032)	F1	0.852 (0.026)	F1	0.816 (0.025)

Table 6.5: Background detection results of original object detection models.

Extended SSD		Extended Faster R-CNN		Extended FCOS	
CLASS	**AP (σ)**	**CLASS**	**AP (σ)**	**CLASS**	**AP (σ)**
Knife	0.715 (0.039)	Knife	0.806 (0.028)	Knife	0.766 (0.035)
Gun	0.894 (0.021)	Gun	0.939 (0.017)	Gun	0.938 (0.022)
Wrench	0.695 (0.053)	Wrench	0.822 (0.027)	Wrench	0.704 (0.025)
Pliers	0.697 (0.044)	Pliers	0.837 (0.032)	Pliers	0.899 (0.029)
Scissors	0.689 (0.038)	Scissors	0.798 (0.035)	Scissors	0.803 (0.029)
mAP	0.738 (0.033)	mAP	0.840 (0.022)	mAP	0.822 (0.027)
F1	0.748 (0.045)	F1	0.831 (0.022)	F1	0.799 (0.039)

Table 6.6: Background detection results of extended object detection models.

Finally, the training and evaluation results of the classification models are shown. Figure 6.2 shows the training graphs of the classification models. All models stopped training early due to an increase in validation loss (indicating overfitting). ResNet, EfficientNet and RegNet stopped after 3 epochs, whereas MNASNet stopped after 5 epochs. All models showed an average training time of about 1 hour, and an inference time of less than one second. Furthermore, all models show stable learning and converge nicely.

As can be seen in Table 6.7, most classifiers outperform previous approaches in background detection performance. More specifically, ResNet, EfficientNet, and RegNet outperform all previous models/approaches in background detection, while MNAS shows similar results to the object detection models in this task. ResNet and EfficientNet perform the best, achieving an average F1-score of 0.957 and 0.953 respectively. RegNet obtains an average F1-score of 0.921, followed by MNASNet with the lowest F1-score of 0.784.

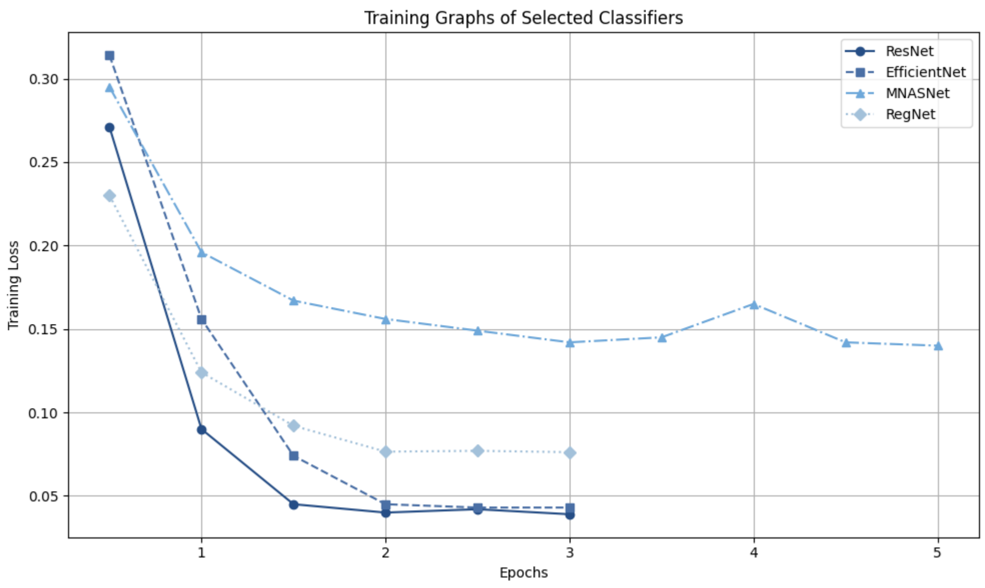

Figure 6.2: Training graphs of the selected classifiers.

Model	Mean Precision (σ)	Mean Recall (σ)	Mean F1-Score (σ)
ResNet18	0.950 (0.025)	0.964 (0.021)	0.957 (0.017)
EfficientNet B3	0.938 (0.023)	0.968 (0.027)	0.953 (0.019)
MNASNet 1_3	0.849 (0.029)	0.728 (0.032)	0.784 (0.025)
RegNet X_1_6GF_V2	0.929 (0.017)	0.913 (0.024)	0.921 (0.022)

Table 6.7: Results of binary classification for background detection.

CHAPTER 7

Discussion and Future Research

In light of the presented results, this chapter discusses the research. It starts by interpreting the main results of the conducted experiments in Section 7.1. Section 7.2 addresses the practical implications of this research, followed by a discussion of the main limitations in Section 7.3. Finally, the chapter will end with recommendations on future research directions in Section 7.4.

7.1 Interpretation of the Results

7.1.1 Freezing Layers During Training

Looking at Tables 6.1 and 6.2, it can be seen that the performance of all models (SSD, Faster R-CNN, FCOS) when freezing layers is not great. Especially, the SSD model, which achieves the lowest average mAP score. Although the Faster R-CNN and FCOS models do not perform that badly, their performance at this stage is not nearly good enough for real-world use.

A possible reason for the poor performance of all models when freezing layers could be related to the dataset used to pretrain the models (COCO). When inspecting the COCO dataset, it can be seen that it contains images that are very different compared to the SIXRay images used for this research. Because of this, information learned during pretraining does not transfer well to the specific task of recognising contraband on X-ray scans. This leaves two options to improve performance: using other pretrained weights, or training all layers. Using weights pretrained on a dataset more similar to SIXRay could infer better transformation of learned information, leading to better performance. However, as mentioned before, similar datasets to SIXRay are not widely available and often classified. Because of this, no pretrained models are available that were trained on a dataset similar to the SIXRay dataset. This leaves only the second option: training all layers. As mentioned, this produced the results of Table 6.3 and greatly improved performance.

7.1.2 Comparison of Object Detection Models

Taking a closer look at the best performing object detection models (Table 6.3), some interesting observations can be made. When comparing single-stage (SSD) and two-stage (Faster R-CNN) models, the most obvious difference is their performance. As previously mentioned, the faster R-CNN always outperforms the SSD model on all metrics by a significant amount. As the Faster R-CNN is considerably more complex, this result is expected. However, an unexpected result is the fact that all models (including FCOS) show a similar inference time of less than one second. Because single-stage methods are known for their speed, it was expected to be significantly faster compared to the other models. While this is the case for model training times, no significant difference in inference speeds was found. This eliminates the expected advantage of single-stage anchor-based models for the specific task of detecting objects in X-ray imagery.

As mentioned in the results, it can be seen that the easiest objects to detect seem to be guns. This is probably because, out of all object classes, guns have the most distinctive shape (always clearly containing an angled barrel and handle). A closer visual inspection of the data also showed a relatively constant size of guns in the data. As explained earlier, the relatively constant shape and size give anchor-based models (SSD and Faster-RCNN) an advantage. This becomes apparent in the results as both the SSD and the Faster R-CNN model outperform the FCOS model in the detection of guns.

The opposite happens when taking the detection of scissors into account. As can be seen, the anchor-based models (SSD and Faster-RCNN) have the most difficulty with the detection of scissors. This is caused by the high variation in shape of scissors compared to other object classes (because scissors can be open or closed and both appear in the data). The variance in shape acts as a disadvantage for anchor-based models, while giving an advantage to more flexible anchor-free models like FCOS. The higher performance in scissor detection of the FCOS model compared to the SSD and Faster-RCNN model reflects this.

7.1.3 Performance of Extended Object Detection Models

As mentioned in the results and shown in Tabels 6.5 and 6.6, extending the object detection models hurts their performance. A probable reason for the decrease in performance is the fact that the models try to learn non-existing bounding boxes for the negative class. As explained in Section 5.1.3, the edges of the entire input image are returned as a bounding box target for negative samples. This is done because the model needs bounding box targets to properly learn. However, these edges do not provide any actual information for the model to learn as there is no bounding box present and no object to detect. The poor learning of the background class bounding boxes hinders the bounding box regression of other (real) classes, resulting in worse results in both object and background detection when compared to the original object detection models.

7.1.4 Comparison of Classification Models

Considering Table 6.7, the results suggest that the ResNet and EfficientNet are the best performing classifiers. The models show no significant differences in training and inference speed, while obtaining a similar F1-score. Taking a closer look at the best performing classifiers reveals a difference in precision and recall. More specifically, the ResNet model has a higher precision and a lower recall when compared to EfficientNet. This means that ResNet makes more accurate positive predictions, making it more efficient compared to EfficientNet as a lower rate of bags will unnecessarily be checked. However, lower recall means that the ResNet model also misses a higher ratio of positive bags, thus making EfficientNet safer. In practice, a trade-off should be made to consider what is more important and a choice should be made accordingly. Nevertheless, both models perform well and show minor differences in results.

7.2 Practical Implications

Taking the results as presented in Chapter 6 into account, it can be stated that the presented models have shown potential to improve the current X-ray screening procedure at airports. As the model pipeline consisting of the best performing binary classifier (ResNet or EfficientNet) and an anchor-based two-stage object detector (Faster R-CNN) produces the best results, this is the proposed approach for real-world use of computer vision in this specific context. As stated, both the object detection and the classification models all have an inference speed of less than one second, making them significantly faster compared to current security operators that perform this task. Furthermore, the proposed model pipeline is not influenced by time pressure, fatigue, and biases, making it more reliable, consistent, and safer. Finally, the model pipeline shows a high level of performance in both the object and background detection task. Because of this, it can be concluded that the proposed model pipeline could replace X-ray screening security operators (classification part of the model pipeline) and aid security operators in the further inspection of positive bags (object detection part of the model pipeline). This results in a more efficient, safer, and cheaper security process that is less reliant on specialised security personnel.

To paint a clearer picture of the practical implications of the proposed model pipeline, a simple example is presented. To further highlight the safety/efficiency trade-off, the same example is considered twice: first using the ResNet model as the pipeline classifier and secondly using EffcientNet.
Consider 1 million pieces of hand-luggage to be checked, 1% of which contain prohibited items (real-world ratio). Using the stated precision (0.950) and recall (0.964) of the ResNet classifier, a simple calculation shows that from these 1 million bags:

- 10148 bags will be checked in total.
- 508 of these bags will be checked unnecessarily.
- 360 bags containing prohibited items will be missed, the other 9640 bags containing prohibited items will be found.

Using the Efficient net (precision of 0,938 and recall of 0.968):

- 10320 bags will be checked in total.
- 640 of these bags will be checked unnecessarily.
- 320 bags containing prohibited items will be missed, the other 9680 bags containing prohibited items will be found.

As predicted, using EfficientNet is less efficient compared to ResNet. EfficientNet checks a total 172 more bags, and checks 132 more bags unnecessarily. However, EfficientNet catches 40 additional bags containing prohibited items, making it the safer option.
No matter which classifier is used, positive predictions get send through to the Faster R-CNN object detection model. With an mAP of 0.873, the model correctly (correct class and location) aids in the further inspection of positively predicted bags 87.3% of the time. Correctly aiding in the identification of the most dangerous object (guns) 94.7% of the time and 81.3% of the time in the identification of the least dangerous object (scissors).

7.3 Research Limitations

Several limitations were observed during the course of this research. First, the reliance on open-source datasets, although necessary for reproducibility, introduces constraints in terms of data diversity. The SIXRay dataset, while extensive, does not fully reflect the variety of prohibited items and complex packing scenarios that might be encountered in airport security checks. An example of this is the absence of a drug or bomb class, two object classes that often need to be considered in the context of airport security. This may limit the generalisability of the models trained on such data to real-world airport settings.
Furthermore, the used data is manually labelled and annotated. Although labels and annotations have a huge effect on the results, details on the labelling and annotation process are not shared. It is not known whether all luggage was opened after being X-rayed to make sure all labelling and annotation was done correctly. If this is not the case, objects (especially those that are hard to detect and thus most valuable in a dataset) could be missed or incorrectly labelled/annotated.

A consequence of this could be that the model has no chance to learn the hardest objects as they are missed when creating the dataset and thus not labelled or annotated. Secondly, models could obtain a wrongfully classified false positive when detecting an object that is actually there but missed when labelling/annotating, resulting in misleading evaluation metrics. A more extensive dataset with a clarified labelling/annotation policy will improve both the results and explainability of this research.

Secondly, deep learning models, especially complex architectures like the object detection frameworks, are often considered "black boxes", meaning their decision-making processes are difficult to interpret. While the models may produce accurate predictions, it is hard to explain how they arrived at those predictions. This lack of interpretability could present challenges in highly regulated industries like aviation security, where transparency is critical.
Additionally, this makes it difficult to draw conclusions when comparing models. Because of the "black box" concept, it is hard to pinpoint which aspects of a model are the reason for bad/good model performance or why certain methods and techniques improve/hurt performance.

Finally, the research only demonstrates the effectiveness of these models in a theoretical environment. Although real-world data has been used, there has been no testing of the models in actual airport settings under real-time conditions. Unforeseen variables such as noise, varying levels of X-ray image quality, and human interactions could all affect model performance. Without real-world testing, the results may not fully reflect the challenges of real-time deployment in busy airport environments.

7.4 Future Research Directions

Building on the findings of this research, several potential avenues for future research emerge. An obvious choice is research that extends open-source datasets in the specific context of X-ray imagery of luggage. As mentioned in Section 7.3, data scarcity is one of the main problems related to this research environment and before computer vision can be adopted in real-world scenarios, research needs to be done to improve this situation. The creation of more extensive datasets has the potential to be very beneficial for the use of computer vision in X-ray screening.

Building on this observation, another option for future research could be the use of generative models to generate synthetic training data that can mimic the complexities of real X-ray scans. As noted, Generative Adversarial Networks (GANs) have shown potential in generating useful synthetic data. In addition to helping solve the data scarcity problem, GANs also have the potential to aid in the training of imbalanced classifiers, another relevant aspect in the context of using computer vision to detect prohibited items in X-Ray scans.

Another potential opportunity for further research relates to the use of different object detection models. This research shows that the anchor-based two-stage Faster R-CNN model outperforms the anchor-based single-stage SSD and anchor-free FCOS model. Based on these results, new research could be conducted comparing different anchor-based two-stage models, further analysing their performance. Another option could be to repeat a similar experiment as the one presented here but with different anchor-based single-stage, anchor-based two-stage, and anchor-free object detection models to see if similar results are obtained. Furthermore, if more extensive datasets become available, the same experiment could be repeated with new object classes to see if the Faster R-CNN model still produces the best results.

Finally, two research proposals considering real-world implications are made. First, studying how object detection models can be made robust against adversarial attacks (intentional modifications made to luggage designed to deceive the models) is important. This would ensure the models remain reliable even in situations where an attacker might attempt to manipulate the inputs.

Secondly, future work could include studies on how the feedback from current security personnel could be used to improve the computer vision models. Introducing a combined operator/model design approach, where the models learn from the corrections and adjustments made by human operators, could solve (parts of) the data scarcity problem and improve existing model performance or make way for new novel approaches.

CHAPTER 8

Conclusion

This paper presents an applied theoretical framework/proof of concept for the use of object detection models for the detection of prohibited items in X-ray images of hand luggage. Through extensive experimentation, this study was able to compare an anchor-based single-stage (SSD), anchor-free two-stage (Faster R-CNN), and anchor-free (FCOS) object detection approach for this specific task on a reliable open source dataset.
Results show that the anchor-based two-stage approach outperforms all other approaches. More specifically, the Faster R-CNN model performed the best in terms of mean average precision, obtaining an average mAP score of 0.873 in the object detection task. The anchor-based single-stage SSD model lagged behind with a much lower average mAP of 0.772. The anchor-free FCOS model, although not as accurate as Faster R-CNN, still offered competitive results (average mAP of 0.843) with a higher degree of flexibility compared to the anchor-based models, outperforming them in the detection of objects with a broader range of shape variations.

Furthermore, this research extends the scope of object detection evaluation by implementing a second evaluation metric: background detection. Including this metric gives a better indication of real world implication of the models. The inclusion of the background detection metric showed that, although (extended) object detection models can produce reasonable results, binary image classification networks performed better in the background detection task while also being computationally cheaper.

Because of this, a model pipeline is proposed consisting of an image classifier and an object detection model. The image classification model separates positive and negative luggage, determining which bags need further inspection. Positive instances are further inspected by the object detection model (only trained on positive data), which does the actual localisation and classification of the detected contraband. The proposed image classifier is either the ResNet or EfficientNet framework, as these achieved the highest average F1-score during testing. When making a final choice, a trade-off between safety and efficiency should be made. As ResNet obtained the highest precision, it has proven to be the most efficient model. On the other hand, EfficientNet showed the highest recall, thus being the safest option. The proposed object detector is the Faster R-CNN framework as this shows the best performance while being fast enough for real-world use.

Taking the presented research into account, it can be concluded that computer vision shows great potential to enhance the passenger security screening process at airports. To be more precise, the results of this research show potential for the use of object detectors and image classifiers to improve the detection of contraband in X-ray imagery. The implementation of said models could be a large step towards the automation of passenger security screening, making it less dependent on human operators, and thus cheaper, while at the same time increasing the efficiency, consistency, and safety of the security process. To conclude, the research question is repeated and answered as shown below:

Research Question: *What performance level can computer vision models obtain in the detection of prohibited items in X-ray images of hand luggage and how do different methods compare?*

Research Answer:*This research has shown that computer vision models have the ability to achieve a high performance level in the detection of prohibited items in baggage X-ray imagery. To clarify further, the best performing object detection model (Faster R-CNN) achieved an average mAP score of 0.873 in the object detection task, while the best performing classifiers (ResNet and EfficientNet) obtained an F1-score of around 0.95 in background detection. Furthermore, the anchor-based two-stage object detection approach outperformed all other approaches. However, the more flexible anchor-free models seem to have a better ability in the detection of objects with varying shapes.*

Bibliography

[1] Samet Akcay and Toby P. Breckon. "An evaluation of region based object detection strategies within X-ray baggage security imagery". In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 1337–1341. DOI: 10.1109/ICIP.2017.8296499.

[2] Samet Akcay et al. "Using Deep Convolutional Neural Network Architectures for Object Classification and Detection Within X-Ray Baggage Security Imagery". In: *IEEE Transactions on Information Forensics and Security* 13.9 (2018), pp. 2203–2215. DOI: 10.1109/TIFS.2018.2812196.

[3] Laith Alzubaidi et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions". In: *Journal of big Data* 8 (2021), pp. 1–74. DOI: https://doi.org/10.1186/s40537-021-00444-8.

[4] Airlines for America. *World Airline Traffic and Capacity*. Accessed: 2024-09-25. 2024. URL: https://www.airlines.org/dataset/world-airlines-traffic-and-capacity/.

[5] Alexandre G. de Barros and David D. Tomber. "Quantitative analysis of passenger and baggage security screening at airports". In: *Journal of Advanced Transportation* 41.2 (2007), pp. 171–193. DOI: https://doi.org/10.1002/atr.5670410204. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/atr.5670410204. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/atr.5670410204.

[6] Herbert Bay et al. "Speeded-Up Robust Features (SURF)". In: *Computer Vision and Image Understanding* 110.3 (2008). Similarity Matching in Computer Vision and Multimedia, pp. 346–359. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.

2007.09.014. URL: https://www.sciencedirect.com/science/article/pii/S1077314207001555.

[7] Neelanjan Bhowmik et al. *The Good, the Bad and the Ugly: Evaluating Convolutional Neural Networks for Prohibited Item Detection Using Real and Synthetically Composited X-ray Imagery*. 2019. DOI: https://doi.org/10.48550/arXiv.1909.11508. arXiv: 1909.11508 [cs.CV]. URL: https://arxiv.org/abs/1909.11508.

[8] David Bremmer. "Schiphol introduceert razendsnelle security: 'Handjes naar beneden en doorlopen'". In: *Algemeen Dagblad* (2024). URL: https://www.ad.nl/binnenland/schiphol-introduceert-razendsnelle-security-handjes-naar-beneden-endoorlopen~acc9f751/.

[9] Rung-Ching Chen et al. "Selecting critical features for data classification based on machine learning methods". In: *Journal of Big Data* 7.1 (2020), p. 52. DOI: https://doi.org/10.1186/s40537-020-00327-4.

[10] James L. Crowley. "Machine Learning with Neural Networks". In: *Human-Centered Artificial Intelligence: Advanced Lectures*. Ed. by Mohamed Chetouani et al. Cham: Springer International Publishing, 2023, pp. 39–54. ISBN: 978-3-031-24349-3. DOI: 10.1007/978-3-031-24349-3_3. URL: https://doi.org/10.1007/978-3-031-24349-3_3.

[11] Demetris Demetriou et al. "Real-time construction demolition waste detection using state-of-the-art deep learning methods; single–stage vs two-stage detectors". en. In: *Waste Management* 167 (July 2023), pp. 194–203. ISSN: 0956053X. DOI: 10.1016/j.wasman.2023.05.039. URL: https://linkinghub.elsevier.com/retrieve/pii/S0956053X23003872.

[12] Mark Everingham et al. "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88 (2010), pp. 303–338. DOI: https://doi.org/10.1007/s11263-009-0275-4.

[13] Carlos Gershenson. *Artificial Neural Networks for Beginners*. 2003. DOI: https://doi.org/10.48550/arXiv.cs/0308031. arXiv: cs/0308031 [cs.NE]. URL: https://arxiv.org/abs/cs/0308031.

[14] David Gillen and William G. Morrison. "Aviation security: Costing, pricing, finance and performance". In: *Journal of Air Transport Management* 48 (2015). Special Issue on Aviation Security, pp. 1–12. ISSN: 0969-6997. DOI: https://doi.org/10.1016/j.jairtraman.2014.12.005. URL: https://www.sciencedirect.com/science/article/pii/S0969699714001537.

[15] Ross Girshick. "Fast R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[16] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pp. 315–323. URL: https://proceedings.mlr.press/v15/glorot11a.html.

[17] I. Goodfellow et al. *Deep learning*. Vol. 1. MIT press, 2016.

[18] Ian Graves et al. "The Role of the Human Operator in Image-Based Airport Security Technologies". In: *Innovations in Defence Support Systems -2: Socio-Technical Systems*. Ed. by Lakhmi C. Jain, Eugene V. Aidman, and Canicious Abeynayake. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 147–181. ISBN: 978-3-642-17764-4. DOI: 10.1007/978-3-642-17764-4_5. URL: https://doi.org/10.1007/978-3-642-17764-4_5.

[19] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 770–778.

[20] Parvinder Kaur, Baljit Singh Khehra, and Er. Bhupinder Singh Mavi. "Data Augmentation for Object Detection: A Review". In: *2021 IEEE International Midwest Symposium on*

Circuits and Systems (MWSCAS). 2021, pp. 537–543. DOI: 10.1109/MWSCAS47672.2021.9531849.

[21] Artur Kierzkowski and Tomasz Kisiel. "Concept of Multi-criteria Evaluation of the Airport Security Control Process". In: *Advances in Dependability Engineering of Complex Systems*. Ed. by Wojciech Zamojski et al. Cham: Springer International Publishing, 2018, pp. 196–204. ISBN: 978-3-319-59415-6.

[22] Artur Kierzkowski and Tomasz Kisiel. "Simulation model of security control system functioning: A case study of the Wroclaw Airport terminal". In: *Journal of Air Transport Management* 64 (2017). Business model innovation in air transport management, pp. 173–185. ISSN: 0969-6997. DOI: https://doi.org/10.1016/j.jairtraman.2016.09.008. URL: https://www.sciencedirect.com/science/article/pii/S0969699716303854.

[23] Alan (Avi) Kirschenbaum. "The social foundations of airport security". In: *Journal of Air Transport Management* 48 (2015). Special Issue on Aviation Security, pp. 34–41. ISSN: 0969-6997. DOI: https://doi.org/10.1016/j.jairtraman.2015.06.010. URL: https://www.sciencedirect.com/science/article/pii/S0969699715000769.

[24] Arthur Knol, Alexei Sharpanskykh, and Stef Janssen. "Analyzing airport security checkpoint performance using cognitive agent models". In: *Journal of Air Transport Management* 75 (2019), pp. 39–50. ISSN: 0969-6997. DOI: https://doi.org/10.1016/j.jairtraman.2018.11.003. URL: https://www.sciencedirect.com/science/article/pii/S096969971830190X.

[25] Shreyas Kolte, Neelanjan Bhowmik, and Dhiraj. "Threat Object-based anomaly detection in X-ray images using GAN-based ensembles". In: *Neural Computing and Applications* 35.31 (2023), pp. 23025–23040. DOI: https://doi.org/10.1007/s00521-022-08029-z.

[26] Ruturaj Kulkarni, Shruti Dhavalikar, and Sonal Bangar. "Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning". In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. Pune,

India: IEEE, Aug. 2018, pp. 1–4. ISBN: 978-1-5386-5257-2. DOI: 10.1109/ICCUBEA.2018.8697819. URL: https://ieeexplore.ieee.org/document/8697819/.

[27] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[28] J. Y. Lettvin et al. "What the Frog's Eye Tells the Frog's Brain". In: *Proceedings of the IRE* 47.11 (1959), pp. 1940–1951. DOI: 10.1109/JRPROC.1959.287207.

[29] S Linnainmaa. "The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors (Doctoral dissertation, Master's Thesis". MA thesis. University of Helsinki, 1970.

[30] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: (2015). Publisher: arXiv Version Number: 5. DOI: 10.48550/ARXIV.1512.02325. URL: https://arxiv.org/abs/1512.02325.

[31] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60 (2004), pp. 91–110. DOI: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[32] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133. DOI: https://doi.org/10.1007/BF02478259.

[33] Domingo Mery, Daniel Saavedra, and Mukesh Prasad. "X-Ray Baggage Inspection With Computer Vision: A Survey". In: *IEEE Access* 8 (2020), pp. 145620–145633. DOI: 10.1109/ACCESS.2020.3015014.

[34] Domingo Mery et al. "GDXray: The database of X-ray images for nondestructive testing". In: *Journal of Nondestructive Evaluation* 34.4 (2015), p. 42. DOI: 10.1007/s10921-015-0315-7. URL: http://link.springer.com/10.1007/s10921-015-0315-7.

[35] Renata F. I. Meuter and Philippe F. Lacherez. "When and Why Threats Go Undetected: Impacts of Event Rate and Shift Length on Threat Detection Accuracy During Airport Baggage Screening". In: *Human Factors* 58.2 (2016). PMID: 26608048, pp. 218–228. DOI: 10.

1177/0018720815616306. eprint: https://doi.org/10.1177/0018720815616306. URL: https://doi.org/10.1177/0018720815616306.

[36] Caijing Miao et al. "SIXray: A Large-Scale Security Inspection X-Ray Benchmark for Prohibited Item Discovery in Overlapping Images". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 2114–2123. DOI: 10.1109/CVPR.2019.00222. URL: https://ieeexplore.ieee.org/document/8953806/.

[37] Stefan Michel et al. "Computer-Based Training Increases Efficiency in X-Ray Image Interpretation by Aviation Security Screeners". In: *2007 41st Annual IEEE International Carnahan Conference on Security Technology*. Ottawa, ON, Canada: IEEE, Oct. 2007, pp. 201–206. ISBN: 978-1-4244-1129-0. DOI: 10.1109/CCST.2007.4373490. URL: http://ieeexplore.ieee.org/document/4373490/.

[38] Mukul Mohan Pande and Pratik Hazare. "Impact Of Information Technology In Enhancing Airport Security". In: *Journal of Namibian Studies: History Politics Culture* 35 (2023), pp. 5048–5059.

[39] C.P. Papageorgiou, M. Oren, and T. Poggio. "A general framework for object detection". In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. 1998, pp. 555–562. DOI: 10.1109/ICCV.1998.710772.

[40] Peng Peng and Jiugen Wang. *How to fine-tune deep neural networks in few-shot learning?* 2020. DOI: https://doi.org/10.48550/arXiv.2012.00204. arXiv: 2012.00204 [cs.LG]. URL: https://arxiv.org/abs/2012.00204.

[41] Alessio Petrozziello and Ivan Jordanov. "Automated Deep Learning for Threat Detection in Luggage from X-Ray Images". In: *Analysis of Experimental Algorithms*. Ed. by Ilias Kotsireas et al. Cham: Springer International Publishing, 2019, pp. 505–512. ISBN: 978-3-030-34029-2.

[42] Ilija Radosavovic et al. "Designing Network Design Spaces". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020, pp. 10428–10436.

[43] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

[44] F Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, 1961.

[45] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386. DOI: https://doi.org/10.1037/h0042519.

[46] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536. DOI: https://doi-org.vu-nl.idm.oclc.org/10.1038/323533a0.

[47] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "(1986) D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol. I, D. E. Rumelhart and J. L. McClelland (Eds.) Cambridge, MA: MIT Press, pp. 318-362". In: *Neurocomputing, Volume 1: Foundations of Research*. The MIT Press, Apr. 1988. ISBN: 9780262267137. DOI: 10.7551/mitpress/4943.003.0128. eprint: https://direct.mit.edu/book/chapter-pdf/2299556/c018389_9780262267137.pdf. URL: https://doi.org/10.7551/mitpress/4943.003.0128.

[48] Li Shen et al. "Deep Learning to Improve Breast Cancer Detection on Screening Mammography". en. In: *Scientific Reports* 9.1 (Aug. 2019), p. 12495. ISSN: 2045-2322. DOI: 10.1038/s41598-019-48995-4. URL: https://www.nature.com/articles/s41598-019-48995-4.

[49] Jacek Skorupski and Piotr Uchroński. "Evaluation of the effectiveness of an airport passenger and baggage security screening system". In: *Journal of Air Transport Management* 66 (2018), pp. 53–64. ISSN: 0969-6997. DOI: https://doi.org/10.1016/j.jairtraman.2017.10.006. URL: https://www.sciencedirect.com/science/article/pii/S0969699717300029.

[50] Petru Soviany and Radu Tudor Ionescu. “Optimizing the Trade-Off between Single-Stage and Two-Stage Deep Object Detectors using Image Difficulty Prediction”. In: *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. Timisoara, Romania: IEEE, Sept. 2018, pp. 209–214. ISBN: 978-1-72810-625-0. DOI: 10.1109/SYNASC.2018.00041. URL: https://ieeexplore.ieee.org/document/8750729/.

[51] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. *Highway Networks*. 2015. arXiv: 1505.00387 [cs.LG]. URL: https://arxiv.org/abs/1505.00387.

[52] Jiaming Su et al. “Research on Airport Security Inspection Process Based on Retention and Regression Model”. In: *ISCTT 2021; 6th International Conference on Information Science, Computer Technology and Transportation*. Mar. 2021, pp. 1–8. ISBN: 978-3-8007-5727-5.

[53] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. DOI: https://doi.org/10.48550/arXiv.1905.11946. arXiv: 1905.11946 [cs.LG]. URL: https://arxiv.org/abs/1905.11946.

[54] Mingxing Tan et al. “MnasNet: Platform-Aware Neural Architecture Search for Mobile”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019, pp. 2820–2828.

[55] Zhi Tian et al. “FCOS: A Simple and Strong Anchor-Free Object Detector”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.4 (2022), pp. 1922–1933. DOI: 10.1109/TPAMI.2020.3032166.

[56] Qian Wang, Neelanjan Bhowmik, and Toby P. Breckon. “Multi-Class 3D Object Detection Within Volumetric 3D Computed Tomography Baggage Security Screening Imagery”. In: *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2020, pp. 13–18. DOI: 10.1109/ICMLA51294.2020.00012.

[57] Qian Wang, Neelanjan Bhowmik, and Toby P. Breckon. “On the Evaluation of Prohibited Item Classification and Detection in Volumetric 3D Computed Tomography Baggage Security Screening Imagery”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207389.

[58] “Weer urenlang wachten in de rij op Schiphol door personeelstekorten”. In: *NOS Nieuws* (2022). URL: https://nos.nl/artikel/2429979-weer-urenlang-wachten-in-de-rij-op-schiphol-door-personeelstekorten.

[59] Wei Wei and Wang Cheng. “Design of Air Passenger Travel Choice Intention Prediction System Based on Deep Learning”. In: *Scientific Programming* 1 (2022), p. 7340552. DOI: https://doi.org/10.1155/2022/7340552. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2022/7340552. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/7340552.

[60] David F Wiley, Deboshmita Ghosh, and Christian Woodhouse. “Automatic segmentation of CT scans of checked baggage”. In: *Proceedings of the 2nd International Meeting on Image Formation in X-ray CT*. 2012, pp. 310–313.

[61] Shifeng Zhang et al. “Bridging the Gap Between Anchor-Based and Anchor-Free Detection via Adaptive Training Sample Selection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[62] Xia Zhao et al. “A review of convolutional neural networks in computer vision”. In: *Artificial Intelligence Review* 57.4 (2024), p. 99. DOI: https://doi.org/10.1007/s10462-024-10721-6.

[63] Zhong-Qiu Zhao et al. “Object Detection With Deep Learning: A Review”. In: *IEEE Transactions on Neural Networks and Learning Systems* 30.11 (Nov. 2019), pp. 3212–3232. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2018.2876865. URL: https://ieeexplore.ieee.org/document/8627998/.

[64] Barret Zoph et al. “Learning Data Augmentation Strategies for Object Detection”. In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, 2020, pp. 566–583. ISBN: 978-3-030-58583-9. DOI: https://doi.org/10.1007/978-3-030-58583-9_34.

Appendix

A1 Object Detection Models

A1.1 Single Shot Multibox Detector (SSD)

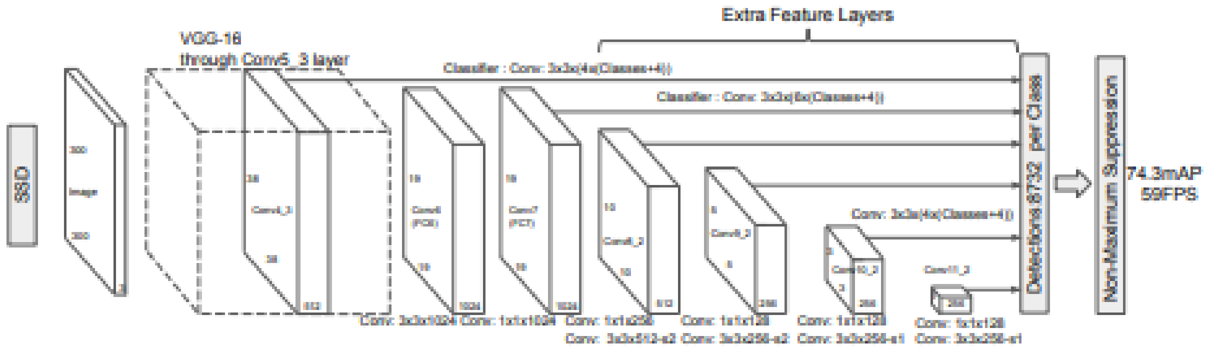

Figure 1: SSD Structure.

A1.2 Faster R-CNN

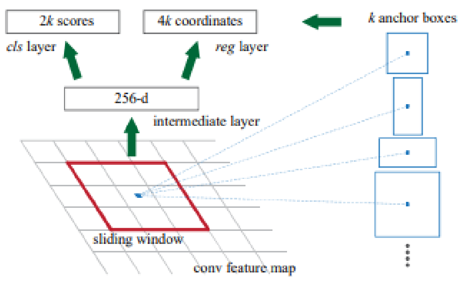

Figure 2: Region Proposal Network Structure.

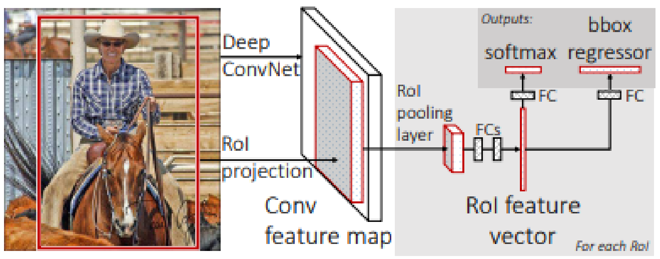

Figure 3: FastRCNN Structure.

A1.3 Fully Convolutional One-Stage Object Detection (FCOS)

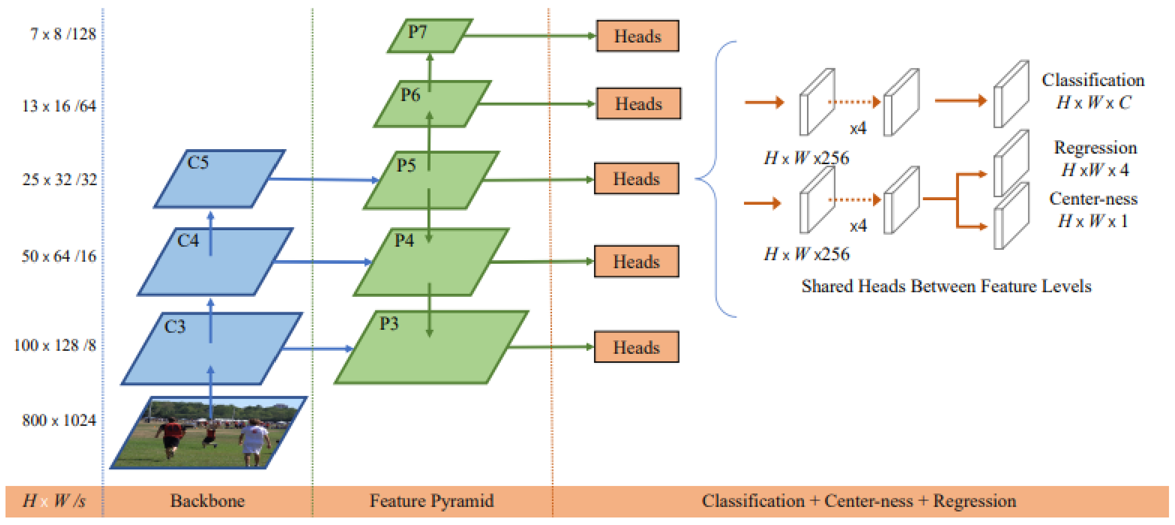

Figure 4: FCOS Structure.

A2 SIXRay Data Samples

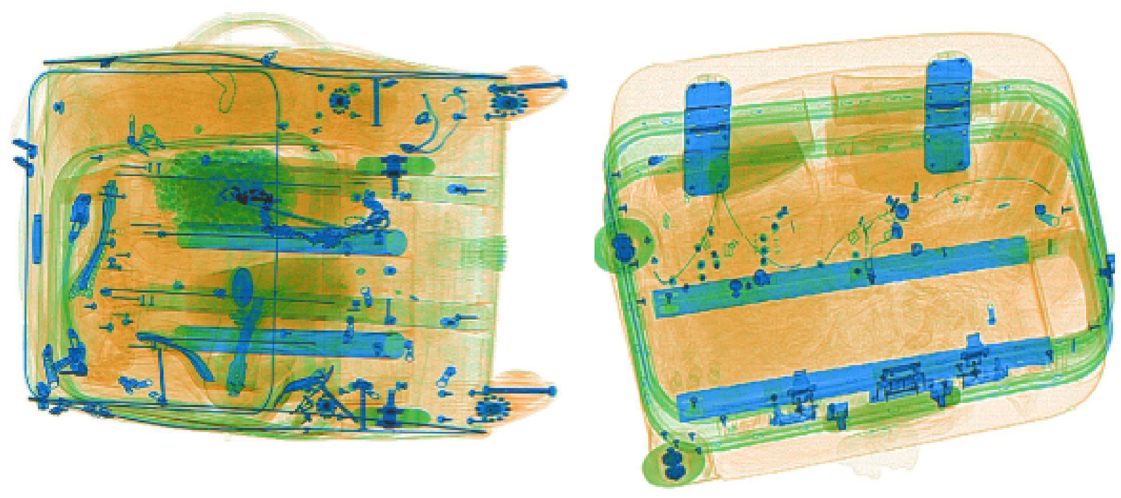

Figure 5: Negative instances.

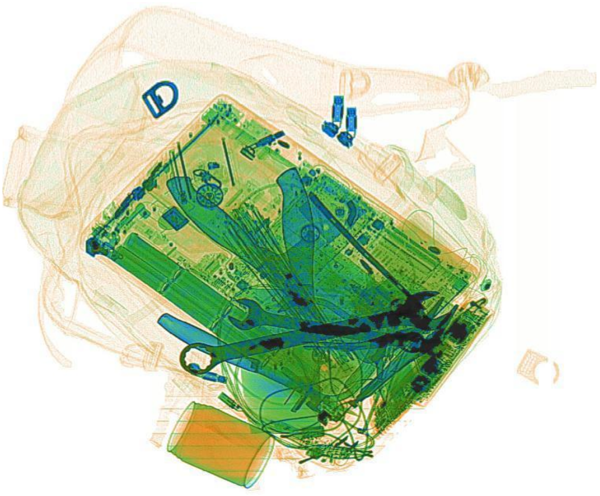
Figure 6: Instance containing a wrench.

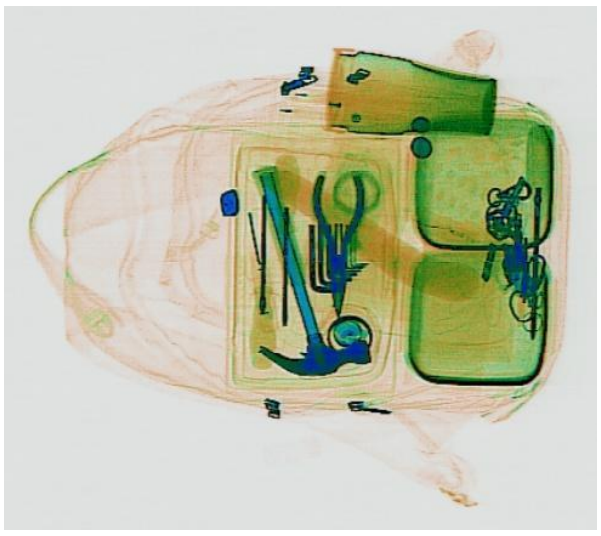
Figure 7: Instance containing a hammer.

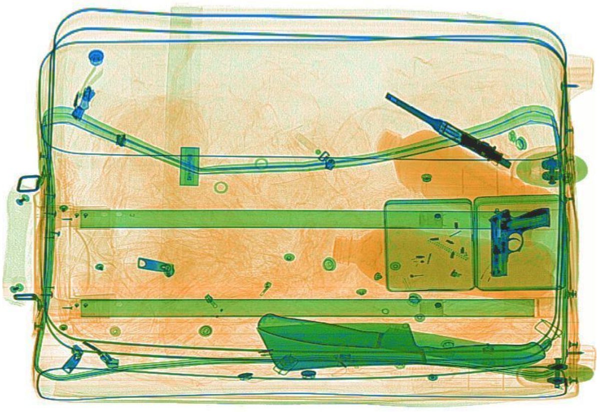
Figure 8: Instance containing a gun.

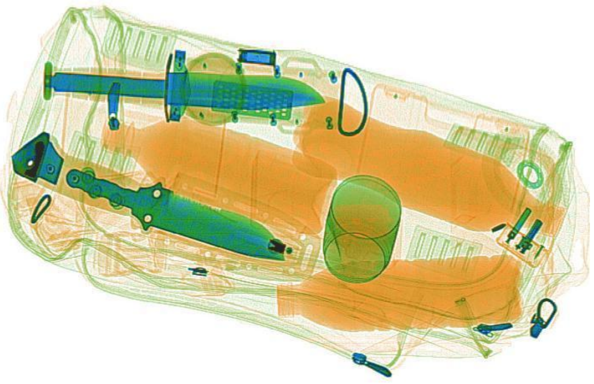
Figure 9: Instance containing a knife.

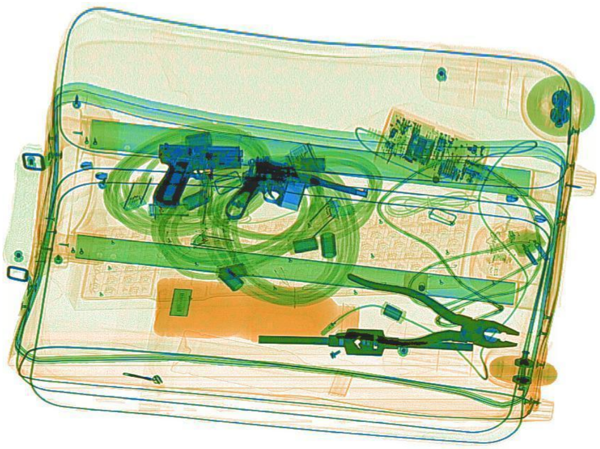
Figure 10: Instance containing pliers.

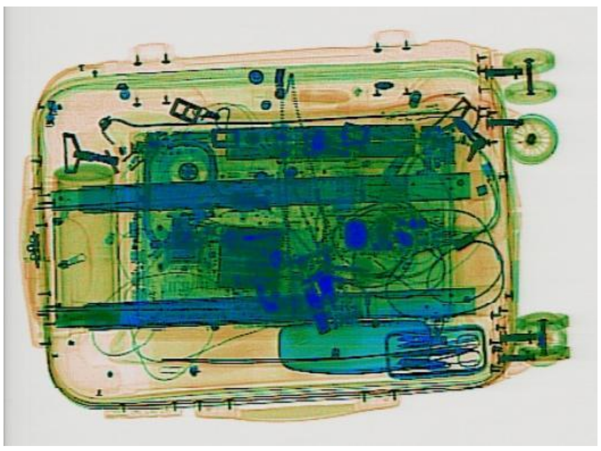
Figure 11: Instance containing scissors.

A3 Data Augmentation

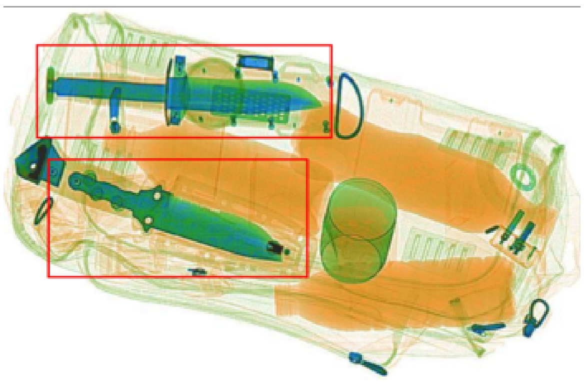
Figure 12: Original image.

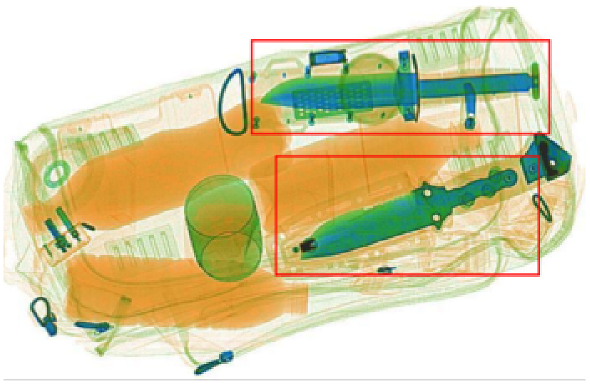
Figure 13: Horizontal flip transformation.

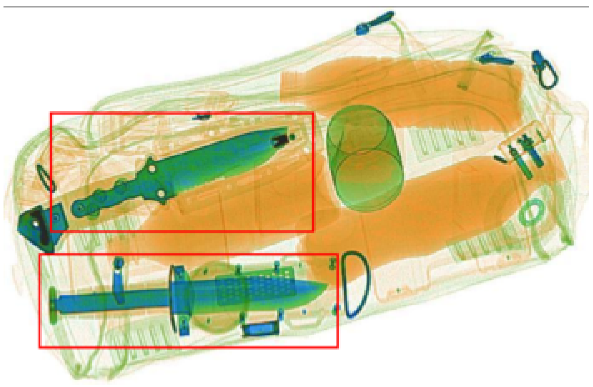
Figure 14: Vertical flip transformation.

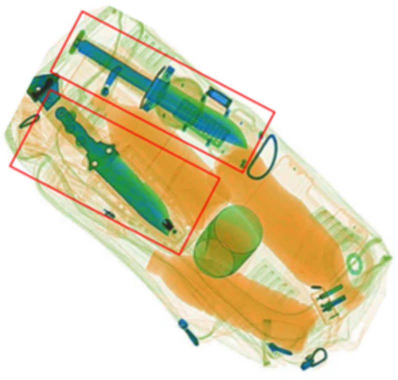
Figure 15: Rotate transformation.

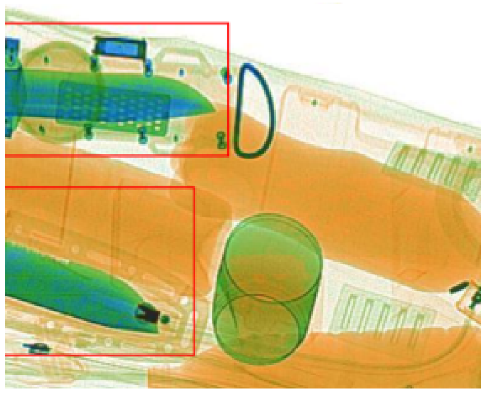
Figure 16: Crop transformation.

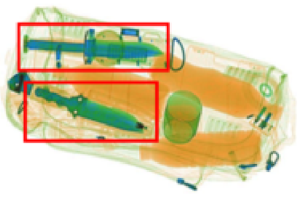
Figure 17: Expand transformation.