Leiden University

ICT in Business and the Public Sector

Automatic classification of handguns in CT images of cabin baggage using Convolutional Neural Networks.

Name:	M.G.A. Rasenberg BSc.
Studentnr:	s1368281
Date:	30/04/2019
1st supervisor:	Dr. A.J. Knobbe (LIACS)
2nd supervisor:	Ir. D.W.J. Meijer (LIACS)
3rd supervisor:	Policy officer (Ministry of Justice and Security)
4th supervisor:	Policy officer (Ministry of Justice and Security)

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Automatic classification of handguns in CT images of cabin baggage using Convolutional Neural Networks.

Mark Rasenberg

Abstract

3D CT imagery of cabin baggage at Schiphol Airport is checked for illegal objects by trained security personnel. This process is labor-intensive as minimum security standards have to be met, but most often no illegal objects are found. Other researchers have suggested machine learning approaches to automatically detect potentially harmful items with promising results, however these classifiers were trained on limited 2D data. In this work, we discuss the possibilities of using 3D data for automated classification of cabin baggage, based on precision at certain recall values. We designed two networks, the first one based on the 2D AlexNet image classification network (Convolutional Neural Network), adjusted to classify 3D data by using 3D kernels, and the second network based on a 2D AlexNet feature extractor feeding its output into two LSTM layers. With our 26,000 3D image dataset, of which 2,500 contain firearms, we were able to get a 90% recall, 76.5% precision, and a 2.7% false alarm rate. This does not outperform the previous CNN based study (Akcay *et al.* [1]) but does outperform previous visual bag of words studies. However due to the different datasets used, the results are not directly comparable. These results meet the minimal requirements of automated software at Schiphol (CONF% recall). It is eligible to be used as an operator assist system, which should result in faster decision times. Considering the upcoming technological progression and the future work for this research there is a big potential towards automated detection of firearms and other potentially harmful items.

Keywords: Automated Classification, Aviation Security, CT imagery, Convolutional Neural Networks, 3D Image Processing

Acknowledgements

First of all, I would like to express my profound gratitude towards my thesis supervisors Dr. Arno Knobbe and Ir. Dirk Meijer, who spend countless hours guiding me towards this final product. With their amazing knowledge of the subject and their fast replies to my questions, this period and the cooperation between us has been a real pleasure.

Next, I would like to thank my supervisors and colleagues at the Ministry of Justice and Security and Schiphol, who gave me the opportunity to work on this amazing topic. They have provided me the required datasets, machine and helped me spread the word of neural networks within the organization. Also, I sincerely thank my acquaintance, who got me the conversation with said supervisors that started this adventure.

I would also like to thank my parents. Mum, dad, thanks for the moral support I have received throughout my educational career. Helping me wherever and whenever necessary and giving me the confidence that I needed to successfully finish my thesis and receive my masters degree. Tessa, thank you for always being interested in what I was doing, being enthusiastic if I was, and flawlessly understanding when I was stressed.

Mark Rasenberg
June 2019

List of Tables

List of Figures

List of Abbreviations

2D	Two Dimensional
3D	Three Dimensional
Adam	Adaptive moment estimation
AISM	Adaptive Implicit Shape Model
AP	Average Precision
CNN	Convolutional Neural Network
CPU	Computer Processing Unit
CSpin	ColorSpin
CS	ClearScan
CT	Computed Tomography
DICOM	Digital Imaging and Communication in Medicine
DICOS	Digital Imaging and Communication in Security
FAR	False Alarm Rate
FATP	False Alarm over the Total Population
FN	False Negative
FP	False Positive
FTE	Full Time Equivalent
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients
ILSVRC	ImageNet large Scale Visual Recognition Competition
LSTM	Long Short-Term Memory
MLP	multi-layer perceptron
NN	Neural Network
PR	Precision Recall

RAM Random Access Memory

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

ROC Receiver Operating Characteristic

SIFT Scale Invariant Feature Transform

SURF Speeded-Up-Robust-Features

TN True Negative

TP True Positive

VRAM Video Random Access Memory

Contents

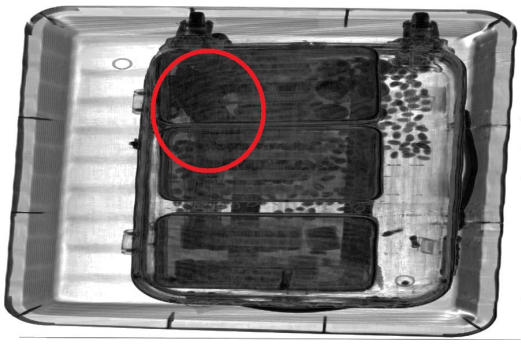
Chapter 1

Introduction

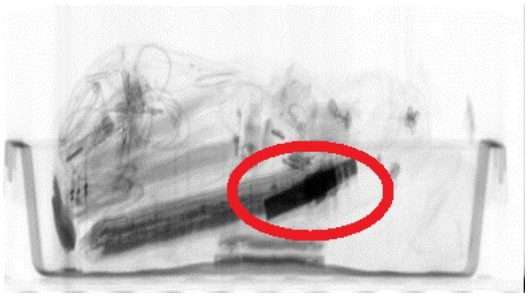
Safety and security are of the highest priority in the aviation industry since airports are a nation's entrance. With over 68.4 million passengers in 2017, Amsterdam Airport Schiphol is the third busiest airport of Europe [10] and the largest airport in the Netherlands. During the 1-week May holiday period in 2018, 5.2 million passengers flew from, to, or via Schiphol. The reason for this passenger spike is that the May holiday period is the same week for everyone in the Netherlands, while there are simultaneously still many business travelers flying. To keep up with this large number of travelers, Schiphol employs additional staff, especially at Security Control. This way Schiphol makes sure passengers go through the airport process as smoothly as possible. Despite the extra security officers, there were still large queues at the check-in, security control, and passport control [11].

Safety and security have such a high priority since security breaches on airports might lead to unwanted items entering the Netherlands or even compromise the security of passengers [12, 13]. This leads to the subject of this thesis, by automating cabin baggage screening the goal is to improve security, and a faster work flow.

This thesis is supported by the National Coordinator for Security and Counter-terrorism (NCTV, [14]). The NCTV protects the Netherlands from threats that could disrupt Dutch society and is responsible for security regulations at Dutch airports. The NCTV graciously provided in association with Schiphol the necessary datasets and the machine that runs the algorithms, these are both further elaborated upon in Section 3.1 and Section 3.2 respectively.



(a) A chaotic x-ray image due to multiple other items which makes it hard to detect the firearm



(b) A bag where the viewpoint of the x-ray machine makes it hard to detect the firearm

Figure 1.1: Two examples of x-ray images where the prohibited items are hard to find, both firearms are circled in red

1.1 Background

Passengers at Schiphol Airport bring two types of baggage, check-in baggage and cabin baggage, which might both contain prohibited objects. Check-in baggage is handed in when you check in for your flight, after which it is moved to a designated area, where it is inspected using large, sophisticated machines and occasionally security officers returned to passengers on arrival. Cabin baggage is different since it is inspected in a large security control hall and returned to the passenger afterwards. The passengers would rather not wait too long to get their cabin baggage back. To keep customer satisfaction reasonably high, the inspection should therefore be fast. The large machines used for check-in baggage are not used in these halls because of space limitations. In these airport halls, the security personnel need to inspect thousands of bags, backpacks, and suitcases daily. These add up to an enormous workload, especially during the holidays, when an larger than normal amount of passengers are going through Schiphol. To make this task more feasible, smaller x-ray machines are used to scan through the baggage, so that fewer bags need to be opened. The output of these machines is checked manually by a trained security officer. Only when an officer finds a potentially harmful item in the x-ray image, the bag is separated and thoroughly checked by another security officer.

There are some shortcomings to the x-ray inspection with these machines. Cabin baggage is often tightly packed, which leads to highly chaotic x-ray images as in Figure 1.1a. Also, items are regularly shown from an unusual viewpoint or are obstructed by other items as in Figure 1.1b. This makes the potentially harmful item interpretation in these x-ray images challenging. Besides this, there are malevolent passengers who intentionally try and hide their belongings from customs. Thus to ensure the images are interpreted correctly, there is a constant need for attention from the security officers.

Tests with new cabin baggage scanners, now used in three out of five security checkpoints, have led to parts of these shortcomings being resolved. The new scanners use CT (Computed Tomography) technologies to generate a 3 dimensional image. The 3D images make it possible to check baggage from multiple viewpoints and look past obstructions [15]. However, these scanners still require a trained security officer to interpret

the images and find potentially harmful objects.

This work explores the automated interpretation of these 3-dimensional cabin baggage images by proposing an algorithm for the automated detection of firearms. We explore solutions to the problem of 3D image interpretation by using state of the art techniques like neural networks. What neural networks are and which methods are used are further explained in Chapter 3.

1.2 Scope of the Thesis

This section will discuss the scope of this work, it will define where the boundaries are set, and why. The main goal is to propose a method, based on neural networks, to try to reduce the queues at security checkpoints, while matching or improving the rate at which potentially harmful items are detected. This improved detection rate is expressed in precision at specific recall values, which is further explained in Section 3.8.

Designing an algorithm based on a neural network includes finding an appropriate dataset. The ratio between baggage containing potentially harmful items and those that do not is very important to achieve adequate, real-world-like results. There are many prohibited items at airports, but since there is only a dataset available containing firearms as described in Section 3.1, this is used as a start to explore the possibilities of automated detection using 3D imagery.

What type of neural network to use and whether it should be combined with some other algorithm will be examined in Section 4. This work will also discuss how the algorithm will be tested, its reliability, in Section 3.8, and to what degree it represents to other items and airports, in Section 6.

With the use of neural networks several implications arise, both in the security and operational aspect of the implementation. We have come up with the following primary research question:

"*What implications arise when using a convolutional neural network on the detection of firearms in cabin baggage at Schiphol Airport?*"

The two implications that our work will discuss are "security effectiveness" and "operational impact" and will be clearly defined in the following paragraphs together with the boundaries of the entire thesis.

The first implication that will be researched is the *security effectiveness* when using a neural network based algorithm. This will be tested by calculating a variety of statistical metrics, mentioned in Section 3.8. The scores of the two types of networks will be compared to each other and the minimum required performance of automated software at Schiphol. According to the results, we will be able to state which one of the proposed methods performs better and whether or not this algorithm performs better or worse than the set baseline. A comparison to the current human performance is not desired due to the confidentiality of these numbers.

The second implication this work will discuss is the *operational impact*. The operational impact is, in this case, described as a combination of the throughput of the system and the staff needed to operate this system. The throughput is equal to the average amount of bags that can be processed per time unit. This part will be somewhat artificial since setting up a real pilot program experiment to find out the actual impact is outside the scope of this thesis. A prediction of the throughput and needed staff will be calculated and compared with the system used currently.

The following topics will only briefly be reviewed in this research:

- *Flexibility of the model*, which is defined as how easy it is to adapt to new inputs in Section 3.4.4.
- *Verifiability of the model*. As of now, the quality of the machines at Schiphol are validated using *a set of test baggage*. However, in this work, the algorithm will be verified using an artificially produced test set in Section 6.4.1.

The algorithm will be mainly based on *convolutional neural networks* (CNN). Besides that, one of the two algorithms uses *recurrent neural networks* (RNN). All of these methods and concepts are further explained in Chapter 3.

1.3 Outline

In Chapter 2, previous studies that were done in this field will be discussed. In Chapter 3 the methods used for the experiments will be explained. This chapter will go through the types of neural networks on which the experiments are conducted and why the parameters are tuned the way they are. In Chapter 5, the risks of using neural networks due to the possibility of adversarial learning are assessed. Next, Chapter 4 will present the results of the conducted experiments. The interpretation of the results are discussed, the shortcomings and future work are discussed in Chapter 6. At last, a conclusion is drawn in Chapter 7.

Chapter 2

Related Work

This chapter discusses the findings of previous studies in the automated aviation security field. A short explanation of the x-ray and CT technologies (which are used to generate the dataset) is given. Then, the previous studies on automated detection of potentially harmful items in cabin baggage will be reviewed in section 2.2.

2.1 X-ray technology

X-ray technology is used for many different applications in the security and medical field. X-ray images can be generated with different energy levels, positions, or dimensions (2D/3D). These generated images can be classified automatically, but in most cases, images recorded by X-ray machines are monitored by specially-trained operators [16, 17]. X-ray technology works with a beam of x-rays, produced by an x-ray generator, being transmitted through an object, for example, the bag of a passenger. The x-rays are then absorbed by the materials from the bag they pass through in different amounts, depending on the density and composition of the materials. X-rays that are not absorbed pass through the object and are recorded with an x-ray detector, which generates the image [18]. This process is visualized in Figure 2.2. There are soft and hard x-rays which respectively have x-rays with lower and higher particle energies. Hard x-rays, with higher particle energy, have better penetration abilities and are therefore used for medical radiography and airport security [19]. With the ability to penetrate through fabric, passengers do not have to open and empty their bags and trained operators can inspect the bags without going through all the bags individually.

Often on airport security, x-ray machines make use of dual-energy x-ray imaging to provide information about material composition or improve image contrast. This technique combines two images acquired at two different energy levels, to obtain both the density and the atomic number of the materials [16, 17, 20].

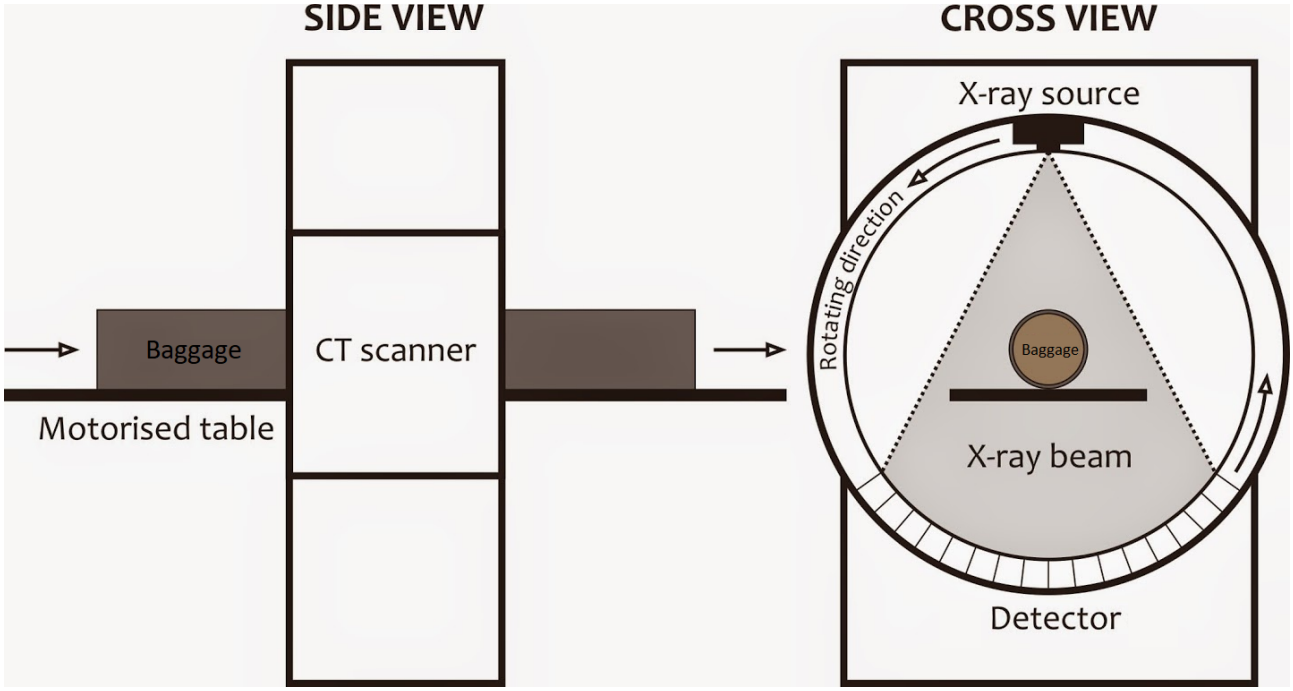


Figure 2.2: A schematic side and cross view of a CT scanner

X-ray images are quite different from the visible spectrum images. First and foremost the images are transparent and its pixel values represent the attenuation of x-ray beams by multiple objects, while objects in the visible spectrum are opaque and occlude each other. Besides that, these images may be very cluttered and noisy due to the low energy x-ray imaging [16,17]. A CT scanner takes away the issue of cluttered images since it produces a 3D image and a noise reduction method that will be addressed in Section 3.7.1.

Figure 2.1: An example of a pseudo colored dual-energy image. Image courtesy of IDSS.

A CT machine does essentially the same as a regular x-ray machine, except for the fact that a CT makes multiple images while spinning around the object. Figure 2.2 represents the CT scanner used at Schiphol. While the baggage moves through the machine, an x-ray source and detector on the other side spin around it generating a 3D image. As pictured a piece of baggage moves through the scanner and while an x-ray source and detector are spinning around it.

The x-ray data are then fed into a tomographic reconstruction program to be processed by a computer. This program deducts the material structure from the x-ray detector values using the Radon transform, a technique first developed by Johann Radon [21]. This results in a series of 2D slices composed into a 3D image with greyscale values based on Hounsfield Units [22], representing density.

The machines that are used at Schiphol combine CT technology and dual-energy technology [23]. A dual-energy x-ray image with little interference is achieved by using two x-ray sources and scanners with an

angular offset of 90 degrees [24]. As a result, the output of these machines is a pseudo-colored 3D image. An example image is found in Figure 2.1.

2.2 Automated detection in cabin baggage

Research on the automated classification of harmful items has been around for some time. The classic methods for detection of items in images are handcrafted features, and will be discussed in Section 2.2.1. The state-of-the-art deep learning methods are discussed in Section 2.2.2. For all research discussed in the next sections, the images in the used datasets were created with dual-energy x-ray machines.

2.2.1 Classic classification methods

In 2011 Baştan *et al.* [16] used a visual bag of words method in combination with point-of-interest detectors and descriptors. The goal was to classify baggage as containing a firearm or not. A shortcoming of their work is the size of the dataset: The training dataset contains approximately 200 images with a firearm and 150 images without. The test dataset is a little bigger with approximately 750 images containing a firearm and 720 without. Both the training and test dataset do not have real-world prior probabilities, where there is rarely a firearm in baggage. The best results were generated with the Scale Invariant Feature Transform (SIFT) descriptor [25]. The results for this setup are recall and precision scores around 0.70 and 0.28 respectively.

In 2012 Franzel *et al.* [26] used a sliding-window object detector with Histogram of Oriented Gradients (HOG) features [27]. A multi-view approach was used, meaning the baggage is scanned from multiple angles. The dataset used in their research was obtained in collaboration with the *German Federal Police* to ensure realism. The dataset contains 770 scans of which 652 contained a weapon, which is a very high prior probability. In this case, it is less of an issue since the research is about detection and not classification. This setup resulted in recall and precision scores around 0.89 and 0.90, which is an improvement over Baştan's experiments.

In 2013 Turcsany *et al.* [28] also used the Bag-of-Words method. The goal of this research was also to detect firearms. There are multiple differences between Turcsany and Baştan, which are mentioned in-depth in [28]. The main differences are the size of the dataset and the type of point-of-interest detector and descriptor. The training dataset that was used contained 10,850 images of which 850 images contained a firearm, which is considerably larger than Baştan's dataset. The test dataset contained 2500 pseudo-colored X-ray baggage images. This research used Speeded-Up-Robust-Features (SURF) [29] for the descriptor. The results for this setup are recall and precision scores of 0.9907 and 0.9569 respectively.

In 2015 Baştan [17] researched a different feature detector and descriptor. The first difference is that the

classification task was expanded to firearms, laptops, and glass bottles. Also, multi-view images were used for the classification task. A larger dataset was used for this experiment, containing 1600 baggage scans which resulted in 6400 images (4 view angle images per scan). Half of the dataset was used for training and the other half for testing the model. The best results in this research were generated with the ColorSpin (CSpin) descriptor [30]. The recall and precision scores for handgun classification are around 0.55 and 0.73 respectively.

In 2015, Riffo *et al.* [31] used the Implicit Shape Model (ISM) [32] and increased its effectiveness and robustness by designing the Adaptive ISM (AISM). This AISM has two main steps: the target object is characterized, then it is detected. A dataset with images of the object to detect from 100 different angles is used to train this algorithm. A category-specific appearance codebook is then generated, with characteristics of the object. In the detection step, features are extracted from the image which needs classification and is compared to the codebook entries. The results of this improved ISM are is a recall score of 0.89 and a precision score of 0.83 for images containing firearms.

In 2016 Mery *et al.* [33] bade a comparison between 6 techniques to classify firearms, shuriken and blades. To compare this technique to the other papers only the results for classifying the firearms are mentioned. All compared methods are displayed in Table 2.1. Each one of these scores is higher on precision and recall than Bag of Words. The relatively low results for the deep learning method might be due to the GDXray dataset [34]. The GDXray dataset does not contain enough images to fully utilize the potential of deep learning methods since only 150 images were used to train the network.

Name	**Recall**	**Precision**
Bag of Words	0.84	0.65
kNN-Based in Sparse Reconstruction Object Recognition	0.99	0.97
Adaptive Implicit Shape Model	0.97	0.97
Adaptive Sparse Representations	0.92	0.88
Deep Learning	0.85	0.99
Baseline Methods	1	0.87

Table 2.1: Recall and precision values for multiple classification methods.

It is worth noting that unfortunately, it is not possible to claim that one method is better than the other since they were not trained and tested on the same dataset and circumstances. The differences in datasets will most likely have influenced the results which are discussed above. Although, according to the individual results of these papers the best classical method is the Bag-of-words method with SURF for the descriptor.

2.2.2 Deep learning classification methods

The first appearance of a convolutional neural network (CNN) used to detect objects in cargo x-ray images is found in research conducted by Jaccard *et al.* [35] in 2016. CNN is further explained in Section 3.5. In this

research, a dataset of 120,000 cargo images was used to train a network to identify cars and 'small metallic threats'. The best performing CNN used a 19-layer structure and achieving a precision of 0.99 for a detection rate of 90%. Even though the classification problem is different because of the fact that the images contain cargo containers, it is relatively similar since the algorithm is also trained to extract features and detect objects using these features.

The latest and most successful research was conducted by Akcay *et al.* [1] in 2018. Our research will be based on and compared to their work, thus this article will be discussed with more depth. Akcay's research consists of multiple sub-parts. First, there is a distinction between the classification task (whether there is a harmful item or not) and the detection task (where the harmful item is located). Second, there are four classification/detection datasets prepared:

1. Dbp_2, this dataset consists of 19.398 manually labeled sub-images, cropped to a 256 x 256 pixel size. These sub-images are cropped from 11,627 2D, dual-energy x-ray images [36]. The dataset has 4,355 positive cases with firearms or firearm components. The classification task conducted in this experiment is the most interesting for this research.

2. Dbp_6, this dataset is designed for the multi-class classification experiment conducted. It consists of the following classes: firearm, firearm component, knives, ceramic knives, laptops. This dataset has the same size as Dbp_2 and are also 2D dual energy x-ray images, but the knives and firearms are sub-categorized.

3. FFOB, this dataset contains 4,680 firearm threat images and 5,000 non-threat images, both 2D dual-energy x-ray.

4. FPOB, this dataset contains 8,770 firearms, firearm parts threat images, and 5,000 non-threat images, all 2D dual-energy x-ray.

For each sample in all datasets, there was random flipping, cropping, and rotation applied to augment the full dataset. However, the augmented samples are kept in the same train-, validation-, or test-dataset.

Each of the experiments used a variety of networks to classify cases. AlexNet [5] was used with weights trained on the ImageNet classification task [37] and layer freezing when training. It turned out the fewer layers were frozen, the better the network performed. Next, both VGG16 [38] and ResNet [39] were experimented with, this time without layer freezing, but again the starting weights were trained on the ImageNet classification task.

They summarize the results of their experiment as:

- *"CNN features achieve superior performance to handcrafted Bag of Visual Words features."*

  (2018, Akcay *et al.* [1])

- AlexNet without layer freezing outperformed the other CNN-based algorithms on the binary classification task (Dbp_2 dataset). It achieved a true positive rate of 99.56% and a precision of 0.997.

2.2.3 Quantification of cabin baggage screening tasks

Wales, A., and Halbherr, T. [40] made an attempt to quantify the cabin baggage screening task. 67 Professional x-ray screeners participated in an experiment of identifying 2048 x-ray images. Half of these images contained threats and the other half did not. The threat images had multiple characteristics:

- Threat categories: firearms/Knives/IED[1]/Other
- Bag complexity: Low/High
- Superposition: Low/High
- View difficulty: Low/High

Unfortunately, due to airport regulations and data protection, no absolute detection values can be reported, which culminates in no usable quantified results.

[1]Improvised Explosive Device

Chapter 3

Methods

This Chapter will describe the datasets in Section 3.1 and the setup that was used for the experiments in Section 3.2. The basic explanation of machine learning, neural networks, CNNs and RNNS are found in respectively Section 3.3, Section 3.4, Section 3.5, and Section 3.6. The experiment setup is found in Section 3.7 and the results of the conducted experiments are validated using the metrics described in Section 3.8.

3.1 Dataset

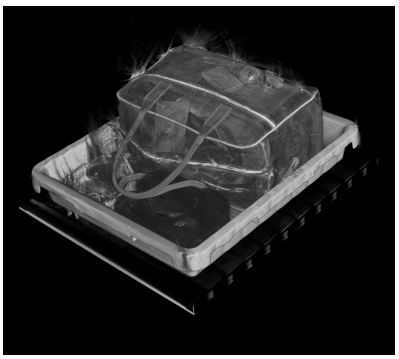
To get an optimal representation of what the algorithm is going to be handling in reality, this dataset only contains images of real cabin baggage. A combination of two datasets is used to train the network. One dataset only contains "positive" images of baggage with, in our case, a firearm while the other dataset only contains "negative" images of baggage, baggage without any potentially harmful items. In Table 3.1, you will find the specifications of each dataset[1]. This dataset also contains some firearms that appear as regular, everyday items.

Table 3.1: Specifications of both datasets, one with the negative samples, and one with the positive samples.

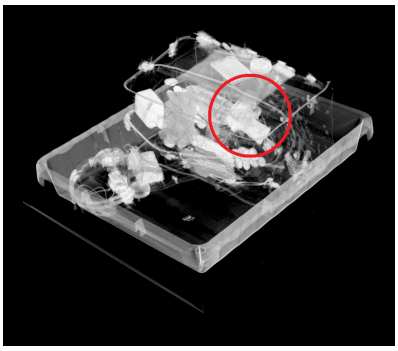
Name	**Positive-dataset**	**Negative-dataset**
Number of images	2,580	26,389
Dimensions of images [x, y, z]	[610, 410, 450]	[610, 410, 450]

The negative-dataset is collected from a cabin baggage-screening CT machine in the field, containing real-life baggage. This dataset is about 91% of the total number of images in both datasets and is checked and labeled by security officers. An example image from this dataset can be found in Figure 3.1c.

[1]The z dimension fluctuates between 380-450. To provide an equal z dimension all images with a z dimension below 450 are padded with zeros.



(a) An example image from the positive-dataset, where the firearm is not visible.



(b) With higher transparency, the contents of a positive bag become visible. The firearm is now visible.

(c) An example image from the negative-dataset, this bag contains no firearm.

Figure 3.1: Examples of both the negative- and positive-datasets.

The positive-dataset was produced in multiple steps. First, isolated firearms were run through the same CT machine as the negative-dataset machine. Then, a different batch of negative images was collected from the exact same machine as the negative-dataset. At last, the isolated firearm images are digitally inserted in the voids of bags. The algorithm is written in such a way that the firearms do not interfere with other objects in the bag. An example image is displayed in Figure 3.1a and 3.1b.

Due to the nature of the dataset, we are unable to use a real world prior probability distribution. The reason is that the percentage of bags with potentially harmful items is very small. For example, if there would only be 3 positive cases per 100,000 bags and the algorithm requires at least 500 positive cases, that would require a dataset of at least 15 million samples. If a dataset with 15 million samples would be available and used, it would increase the computation time of the algorithm drastically. Also, if there are only 100,000 images available, the algorithm has to learn on 3 positive cases. When training on batches of images, it is desirable that each batch has positive cases for the neural network to successfully learn. For these two reasons, oversampling is used in our dataset, resulting in a higher percentage of positive cases to the total number of cases.

3.2 Setup

To run the algorithms that were experimented in this work the following setup has been used:

- CPU: Intel Core i5 8600 (6 cores, 6 threads, 3.1GHz)
- GPU: 2 x Nvidia GeForce GTX 1080Ti
- RAM: 32GB DDR4 (2.4GHz)

Neural networks use many matrix multiplications to calculate the outcome of the network and are faster

when computed on GPUs instead of CPUs [41]. To ensure the confidentiality of the datasets, the setup was disconnected from the internet.

The CNNs in this research were built and ran with the following software:

- Linux, Ubuntu version 16.04.6
- CUDA, version 9.0.176
- Python, version 3.5.2
- Keras, version 2.2.4, [42]
- Tensorflow, version 1.12.0, [43]
- Numpy, version 1.16.3, [44]
- Scipy, version 1.2.1, [45]

3.3 Machine Learning

Machine learning is a term which has been around for some time. Its popularity continues to increase as more opportunities are being discovered. A machine learning algorithm is an algorithm that is able to learn by training with data. In 1997, Mitchell [46] defined machine learning as:

> *"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E"*
>
> (1997, Mitchell [46])

Nowadays, "experience E" in the form of data, is collected almost everywhere, and its availability is higher than ever. From here on "experience" is referred to as *training data*.

For example, a "task T" can be classification (where the category of a sample is predicted), clustering (which is about grouping samples that are alike), or regression (which is predicting some value in a sequence or finds the underlying function). This research defines a classification task T to distinguish cabin baggage with and without potentially harmful items from each other. This task is to classify samples of two classes and is called a 'binary classification' task. The classification task consists of a prediction function that calculates predictions, Equation (3.1). For each of the predictions, the error of the network is calculated using the cost function, Equation (3.2). The prediction function $f(x_i)$ is defined as the prediction of label y_i based on the input x_i of sample i. The cost function calculates the difference between the predicted label $\hat{y}_i$ and the true

label y_i of sample i. The performance measure P returns the arguments of $f(\cdot)$ in which the cost function is minimized.

$$Prediction\_Function : f(x_i) = \hat{y}_i \tag{3.1}$$

$$Cost\_Function : \epsilon = \sum_{i=1}^{N} ||f(x_i) - y_i||^2 \tag{3.2}$$

In machine learning there are multiple ways to train an algorithm to perform a task:

- Unsupervised learning, where the training data is not labeled.
- Supervised learning, where the training data is completely labeled.
- Semi-supervised learning, where the training data partially labeled.

This research is based on semi-supervised learning since the negative dataset is unlabeled and the positive dataset is labeled. We could argue that it is supervised learning for the reason that we assume that the negative dataset does not contain any potentially harmful items, while there is a slim chance that it does.

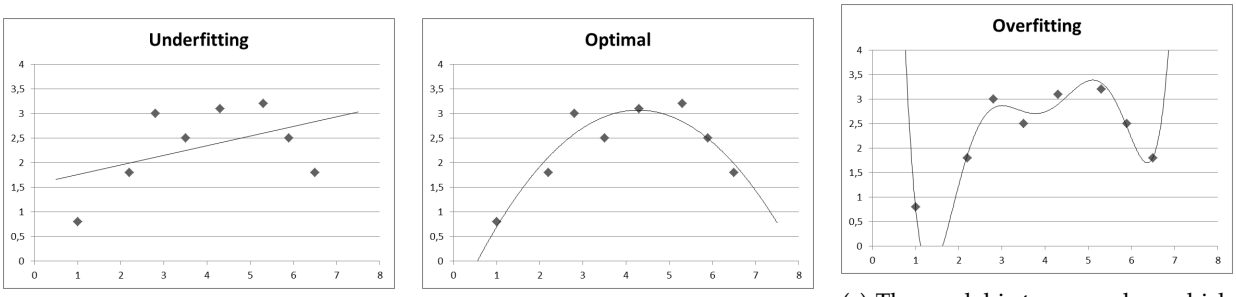
There are a few other concepts from machine learning which need to be discussed before we can broach the neural network topic, which are over-, underfitting, and the curse of dimensionality. These topics will be covered in the next two subsections.

3.3.1 Over- and underfitting

Over- and underfitting are two central challenges in machine learning. Overfitting indicates that the model is trained too exactly to particular training data and is therefore unable to reliably generalize samples outside this dataset. Overfitting occurs when there is too much noise in the training data, there is not enough training data, or the model is too complex[2].

Underfitting is the opposite of overfitting and indicates that the algorithm is unable to sufficiently minimize the performance measure on the training set. It might occur when the features of samples do not contain enough information, when there are not enough samples to train on, or when the model is too simple[3]. An image of both underfitting and overfitting are displayed in respectively Figure 3.2a and Figure 3.2c. With a proper model, results like Figure 3.2b should be produced, making good predictions.

[2]The model is too complex relative to the amount of noisiness of the training data and memorizes too much of the details
[3]The model is unable to learn the underlying structure of the data



(a) Due to simplicity of the model, it is unable to understand the underlying structure.

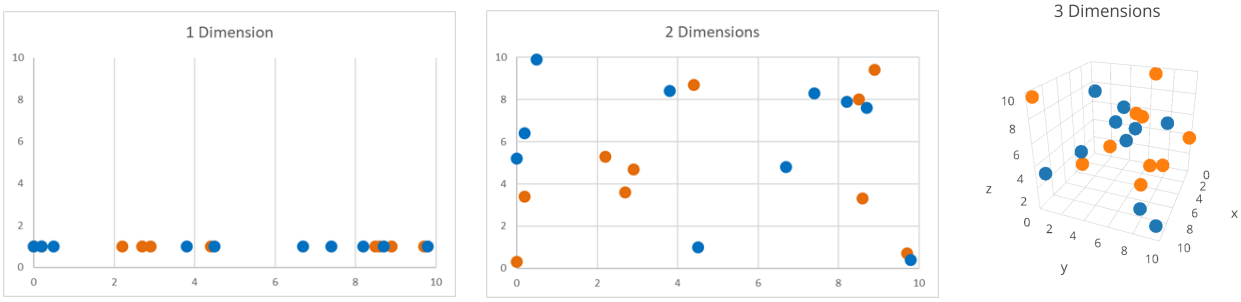
(b) The optimal model represents the underlying structure of the data without fitting noise.

(c) The model is too complex, which lead to the noise in the data to be fitted. Thereby, the model is unable to generalize to new examples.

Figure 3.2: Visualizations of under- and overfitting.

3.3.2 Curse of dimensionality

A big difference between this research and Akcay is the dimensionality of the data. Akcay made use of a dataset with only 2D images, while the dataset used for the experiments conducted in this research contains 3D x-ray images. Unfortunately, the higher dimensionality of the images bring some challenges according to the *curse of dimensionality*. This so-called "curse", has to do with the increasing amount of regions that exist in images that contain information, as pictured in Figure 3.3. This Figure shows that with every dimension added, the number of regions that needs to be considered, rises exponentially. The number of dimensions are in this case equal to the number of voxels in the input images. With the high number of voxels, the time it takes to train the algorithm significantly increases. To reduce the time it takes to train, the algorithm will be trained on a random subsample of the full dataset when fine-tuning the hyperparameters. The final test results will be based upon the full dataset. More information about the AlexNet algorithm and the adjustments can be found in Section 3.7.2.

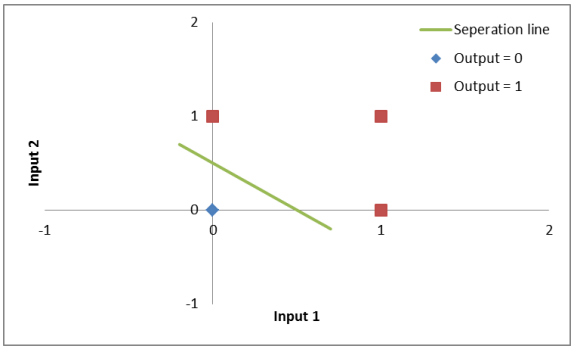


(a) In 1 dimension there are 5 regions to consider.

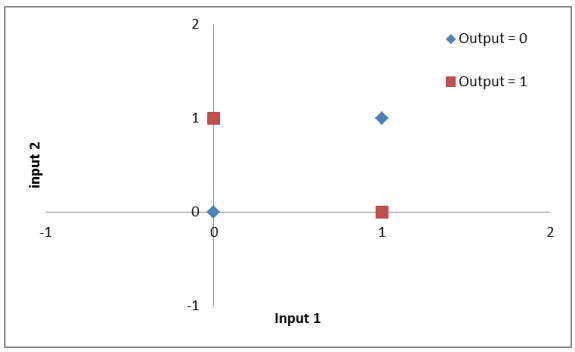
(b) In 2 dimensions there are 25 regions (5^2) to consider.

(c) In 3 dimensions there are 125 regions (5^3) to consider.

Figure 3.3: A visualization of the "curse of dimensionality".



(a) Visualization of a single layer perceptron dividing up the x's and o's 2D vector space, provided by [3]



(b) Visualization of the XOR problem. There is no single straight line that is able to divide the x's and the o's, provided by [3]

Figure 3.5: Dividing vector space

3.4 Neural Networks

As mentioned in Chapter 2, a neural network is the preferred machine learning method for image recognition and is therefore used in this research. This Chapter explains the necessary basics of neural networks to understand this research.

3.4.1 The perceptron

To fully explain the working of a Neural Network, we need to start at the beginning, with the perceptron. The perceptron is a neural network in its most basic form, invented by Dr. Frank Rosenblatt [2] in 1958. Rosenblatt's perceptron is based on the model of a single neuron in a brain and is represented in Figure 3.4.

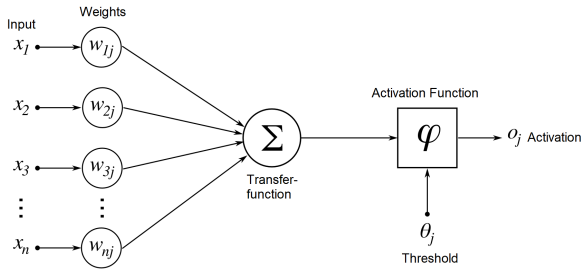


Figure 3.4: A visual representation of a Rosenblatt's Perceptron [2]

The perceptron has a series of inputs (x_n), corresponding weights (w_{nj}) and one output or activation (o_j). There is a transfer function, which is the summation of the input values times the weights up as in Equation (3.3). The result of the transfer function is then put in an activation function, which results in activation, when the threshold (θ_j) is met or no activation (o_j). Rosenblatt's perceptron is an example of a single layer perceptron. This perceptron is able to draw a single line in 2D vector space, shown in Figure 3.5a.

$$\sum_{j=1}^{n} x_j w_j \tag{3.3}$$

Due to this fact, the perceptron is unable to replicate an XOR-function [47]. The XOR-function is an operation on two binary values x_1 and x_2. When exactly one of these values is equal to one, the network should return 1 otherwise 0. The XOR function is shown in 3.5b, not that there is no single straight line that is able to divide the o's and x's successfully solving the XOR function.

3.4.2 Backpropagation and the multi-layer perceptron

Backpropagation is an algorithm which enables the possibility of adapting network weights and biases according to the cost function, first used by Rumelhart *et al.* [48] in a multi-layer perceptron. The backpropagation algorithm works according to a few steps. First, a training sample is fed to the network and the output values of all nodes are computed. Then, the cost function calculates the error resulting from the network (ϵ), the Equation (3.2).

The next step is the backward pass, this is where the changes to the weights that need to be made to minimize the cost function are calculated. The backward pass is executed for all training samples i. $\frac{\partial \epsilon}{\partial \hat{y}_i}$ is calculated for a particular sample i. With the chain rule $\frac{\partial \epsilon}{\partial x}$ can be calculated with Equation (3.4).

$$\frac{\partial \epsilon}{\partial x} = \frac{\partial \epsilon}{\partial \hat{y}_i} \cdot \frac{\mathrm{d}\hat{y}_i}{\mathrm{d}x} \tag{3.4}$$

In Equation (3.4), the value of $\frac{\mathrm{d}\hat{y}_i}{\mathrm{d}x}$ is the derivative of $\hat{y}_i$ with respect to its input. This means we know a change in the input x will affect the error on these samples in a certain direction. Now the derivative with respect to the weights w_i is calculated with $\frac{\partial \epsilon}{\partial w_i}$, as in Equation (3.5).

$$\frac{\partial \epsilon}{\partial w_i} = \frac{\partial \epsilon}{\partial x} \cdot \frac{\partial x}{\partial w_i} \tag{3.5}$$

The change of the weights, Δw_i, is then defined by multiplying the error gradient with the learning rate, α, as in Equation (3.6).

$$\Delta w_i = -\alpha \frac{\partial \epsilon}{\partial w_i} \tag{3.6}$$

With the output of layer $(n-1)$ being the input of layer n, we can state that the desired input states of layer n are the desired output states of layer $(n-1)$. Now, for each previous hidden layer, the change of weights according to the desired input states of the next layer is calculated using $\frac{\partial \epsilon}{\partial x_j}$. This process is then repeated for all layers until the input layer is reached. This process is repeated for all training samples in the training set.

With the use of a multi-layer perceptron (MLP) in combination with backpropagation Rumelhart *et al.* [48] was the first to overcome the limitation of earlier single- and multi-layer perceptron, by being able to solve the XOR problem. A multi-layer perceptron is a combination of multiple perceptrons. The activation of one perceptron becomes the input of perceptrons in the next layer. This structure is shown in Figure 3.6. Until Rumelhart's research on backpropagation, there was no way to train the multi-layer perceptrons. With the XOR-problem solved, neural networks regained their popularity, up to the point where we are now.

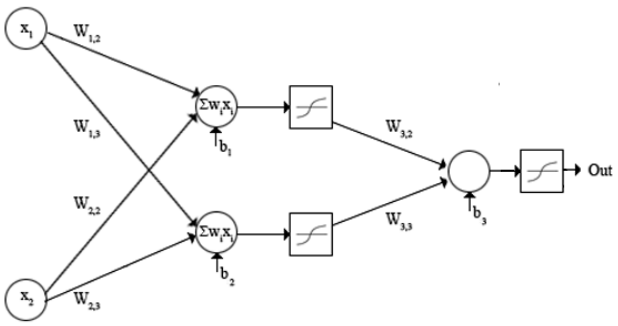


Figure 3.6: A multi-layer perceptron which is able to solve the XOR problem, provided by [3].

Now that the basics of Neural Networks are covered, more in-depth subject can be explained to fully understand the working of the final algorithm in the following subsections:

- Different types of layers in a neural network, Section 3.4.3.
- Difference between batch and online learning, Section 3.4.4.
- The activation and cost functions, Section 3.4.5.
- The gradient descent algorithm, Section 3.4.6.
- The learning rate and Dropout, Section 3.4.7.

3.4.3 Neural Network layers

The standard architecture of a neural network is built with layers which contain single processing unit (for example a perceptron) called "nodes". These layers and nodes are displayed in Figure 3.7. Each node is represented by a circle and contains a single perceptron. Each arrow represents a connection from the output of a node to the input of one node in the next layer. A network starts with a number of input nodes in the input layer. When the network is being trained, the input layer receives the training data. If the network is already trained and deployed, the input layer receives the data that needs to be classified. Then there are one or more hidden layers, whose input nodes take the output of the layer preceeding. Different hidden layers are used to discover higher or lower order patterns. Low-order patterns can be something like a line, while higher order patterns might represent the trigger of a firearm. At last, there is the output layer, which contains the prediction of a class according to the calculations done by the network.

In the case of image processing, the input layer of the network is shaped by the number of pixels in an

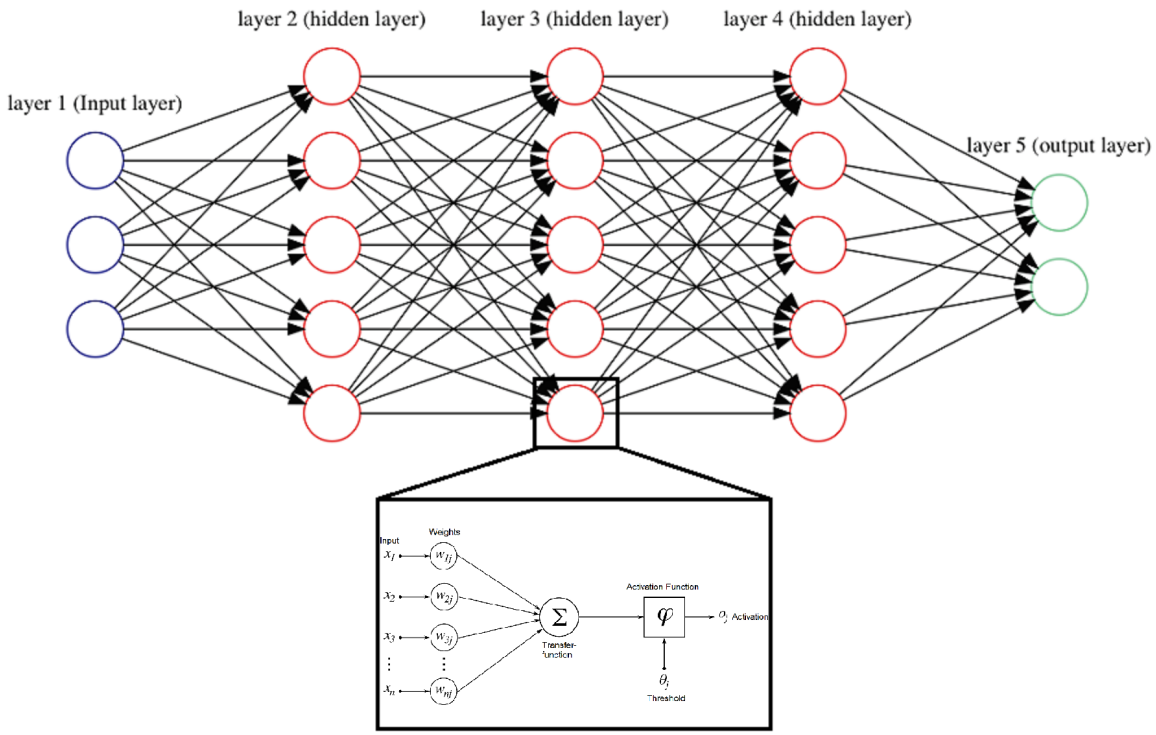


Figure 3.7: A neural network with three hidden layers, multiple input and output nodes.

image. The output layer, on the other hand, is shaped based on the number of possible classes. For example, when the network is trained on 128×128 pixel images, the number of input nodes will be $16,384$. When the network needs to classify which of the x possible classes the data belongs to, the number of output nodes will be equal to x or $x - 1$.

The output node will have a value between 0 and 1. Based on a threshold, θ, the output node will result in a positive or negative classification. The shape and number of hidden layers are based on what the best performance of a network is for a specific dataset.

3.4.4 Batch and online learning

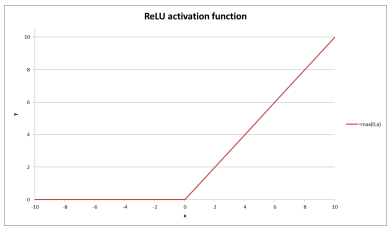
There are two methods on how to teach a network, batch (offline) or online learning. Batch learning implies that the system is unable to learn as an ongoing process. To train the network on newly available data a new network needs to be trained while it is not classifying samples. Online learning, on the other hand, is able to train on new data while it runs. The network is able to adjust its weights according to the new data and adapts faster. Online learning networks are great for systems that continually receive data and need to adapt to change rapidly, but batch learning systems are more robust since the hyperparameters can be tweaked to the newly available data.

3.4.5 Activation and cost functions

The activation function is briefly explained in Section 3.4.1 as the function that defines the output given an input-set and a certain threshold. There are many different activation functions available to use. According to Akcay's paper [1] AlexNet performed the best, which leads to the Rectified Linear Unit (ReLU) activation function to be used in this research' hidden layers and the Sigmoid activation function in the output layer.

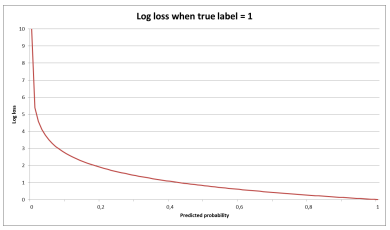
$$f(x) = \max(0, x) \tag{3.7}$$

The ReLU activation function equation is quite a simple function, is displayed in Equation (3.7) and visualized in Figure 3.8a. According to [49] training deep neural networks is better with the ReLU function. Ramachandran *et al.* [50] states that to the date of writing ReLU is the most used activation function, due to newer functions not being able to produce consistent gains.



(a) Visualization of the ReLU activation function

(b) The sigmoid activation function compressing values to the range [0,1]



(c) Cross entropy loss returns an exponentially higher loss value when the prediction is worse

The sigmoid activation function compressed the output of a node to values between 0 and 1. It utilizes the sigmoid Equation (3.8) and the corresponding graph is displayed in Figure 3.8b.

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{3.8}$$

The cost function calculates how much the predicted values were off and is the function that needs to be minimized by the optimizer. There are many different cost functions, which all use different calculations. The most used cost functions for CNN tasks like the one in this research is binary cross entropy. This function is designed for binary classification tasks and penalizes bad predictions much more as displayed in Figure 3.8c and Equation (3.9).

$$H_b(p) = -p \log(p) - (1 - p) \log(1 - p) \tag{3.9}$$

3.4.6 Gradient Descent

Gradient descent [51] is an universal optimizer capable of finding optimal solutions to a range of problems. It is the basis of many other optimizer strategies noted in Section 3.5.6. The gradient descent function moves in the direction yielding the maximum decrease. This direction is calculated by moving in the opposite direction of the local derivative ($f'(x)$). The derivative itself is approximated by adding a small value ϵ to the input x and obtaining the corresponding change in the output, displayed in Equation (3.10).

$$\text{For small values of } \epsilon: \qquad f(x+\epsilon) \approx f(x) + \epsilon f'(x) \tag{3.10}$$

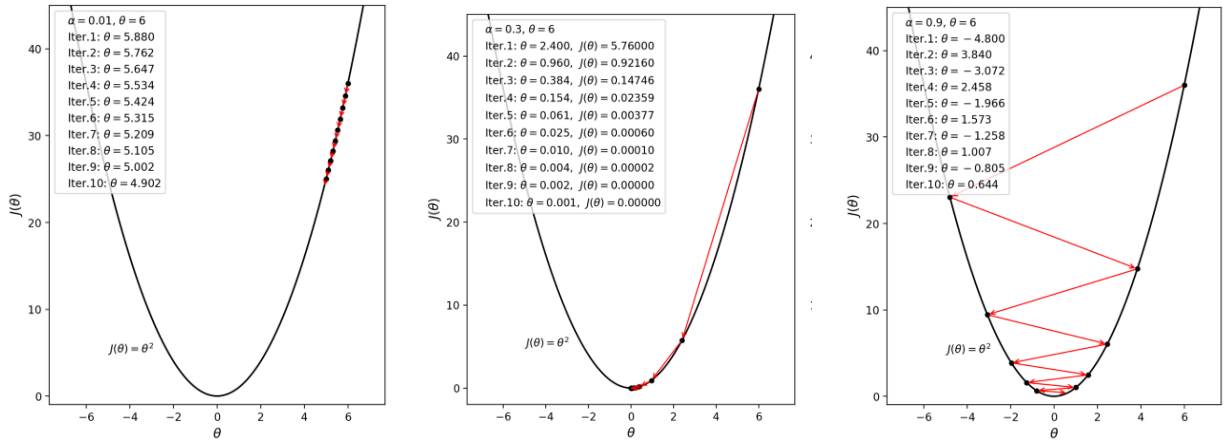
$f'(x) = 0$ when a critical point[4] has been reached. A critical point may be local or global. Local critical points may not lead to the optimal solution, which is the global critical point. Generally, some local critical points are acceptable halting points when the model is performing nearly as well as the global one would have. To overcome the issue of local non-optimal critical points, the learning rate can be adjusted. The learning rate is further explained in the next Section.

3.4.7 Learning rate and dropout

In Section 3.3 and Section 3.4.6, overfitting was mentioned. To reduce overfitting, the learning rate can be adjusted or dropout can be introduced. The learning rate is the parameter which determines how much the network adjusts to the backpropagation step. It decides the size of the step in the direction that yields the maximum decrease. A learning rate that is too small will make the algorithm run many iterations which will take a long time and might even make the algorithm converge at a local optimum, resulting in a model that underfit the data. When the learning rate is set too big, the algorithm can overshoot the global optimum and might not converge at all. A learning rate that is too small, optimal and too big are displayed in respectively Figure 3.9a, Figure 3.9b, and 3.9c. In these figures theta, θ, represents the parameter set of the model (weights and biases). $J(\theta)$ represents the output of the model with a certain parameter set θ.

Another method to reduce overfitting is dropout, which was proposed by Hinton *et al.* [52] and further detailed by Srivastava *et al.* [53]. Dropout is computationally inexpensive and is highly successful and a rather simple algorithm [53]. The algorithm randomly selects nodes with probability p and overwrites the output values of those nodes to 0, deactivating them[5]. This method ensures that the network is able to produce valid results even if nodes are deactivated. No outcome of the network relies on only one path through the network, thus resulting in a more robust network. Dropout is only used while training the

[4] Minimum, maximum, or saddle point
[5] The values resulting from output nodes remain untouched



(a) A learning rate that is too small will take significantly longer to converge

(b) An optimal learning rate gradually approaches the minimum and quickly converges in a (near-) global optimum.

(c) A learning rate that is too big can overshoot the global minimum, taking a large amount of time to converge.

Figure 3.9: Three examples of learning rates that are too small, too big or just right.

algorithm, a trained algorithm uses all nodes.

3.5 Convolutional Neural Networks

Multi-layer neural networks are computationally inefficient when it comes to image classification tasks due to the large number of input nodes. For that reason, convolutional neural networks (CNN) increased in popularity for the last years. CNN's use weight-sharing to reduce the computational complexity [54]. Because of this, CNN algorithms are able to achieve high performance on complex visual tasks, with the ability to process more data in the same amount of time and recognize more complex patterns.

3.5.1 Convolutions

In machine learning applications, the input and the kernel are usually a multidimensional array of data, in our case an image. The convolution operation is used to produce a third array, which expresses how the shape of the input image is modified by the kernel. The 2D discrete convolution operation is typically denoted with an asterisk and the formula is found in Equation (3.11). In this equation, I is referred to as the input image with dimensions (i, j), and K is the filter. The convolution operation iterates from $-\infty$ to ∞ summing up overlapping values of K and I [55].

$$S(i,j) = (I * K)(i,j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I(i-m, j-n)K(m,n) \tag{3.11}$$

The dimensions of S, in every dimension, are equal to the dimensions of I added to the dimensions of K minus 1. Often in deep learning the images used are much larger than the kernel used, $\|I\| \gg \|K\|$, you can state $\|S\| \approx \|I\|$. We force $\|S\| = \|I\|$ by using padding, which is an edge condition and adds an amount of data to the input image to preserve the size. The preservation of dimensions is desired, since the filter size is otherwise limited. An example is given in Figure 3.10, in this image, padding is mandatory since the convolution is unable to calculate the corner values since the kernel starts partially outside of the input image I. By adding 0's to the borders of the image, the convolution successfully calculates the output values while retaining the same dimensions of the input.

3.5.2 Convolutional layers

When working with images, input nodes of a CNN represent the pixels of the input image. A CNN is named after the use of at least one convolutional layer in its architecture. A convolutional layer consists of filters (convolution kernels), that slide across the input image. The example filter in Figure 3.11, consists of a vertical line, which can represent the border of an object in an image. The filter slides left to right, top to bottom. The ability of the filter to find the contrast difference is visible in the featuremap, with higher values on the left side. The filter slides across the image, multiplying its values with the pixel values of the image. This results in a *featuremap*, containing information of where possible vertical lines are detected. Since weights are shared between nodes of a filter, the convolutional layer uses fewer parameters than fully connected layers.

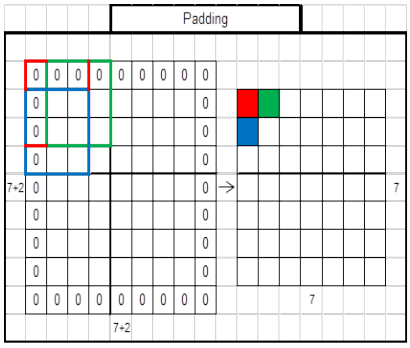


Figure 3.10: This example uses "same" padding.

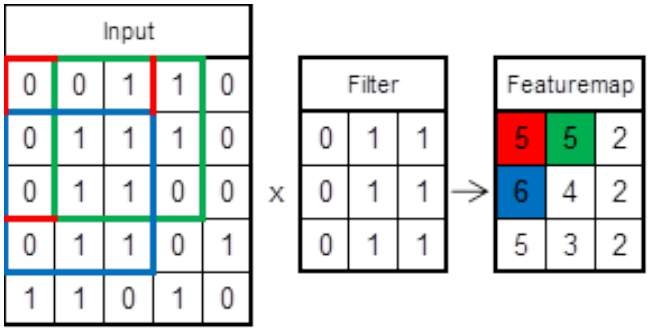
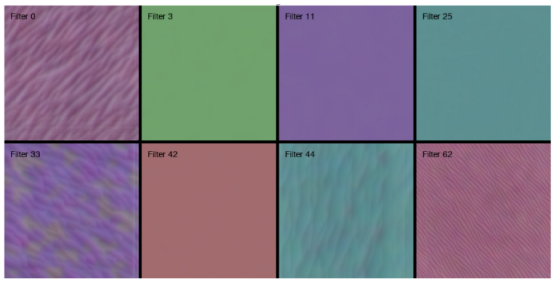
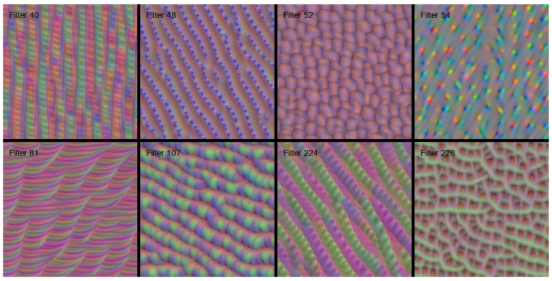


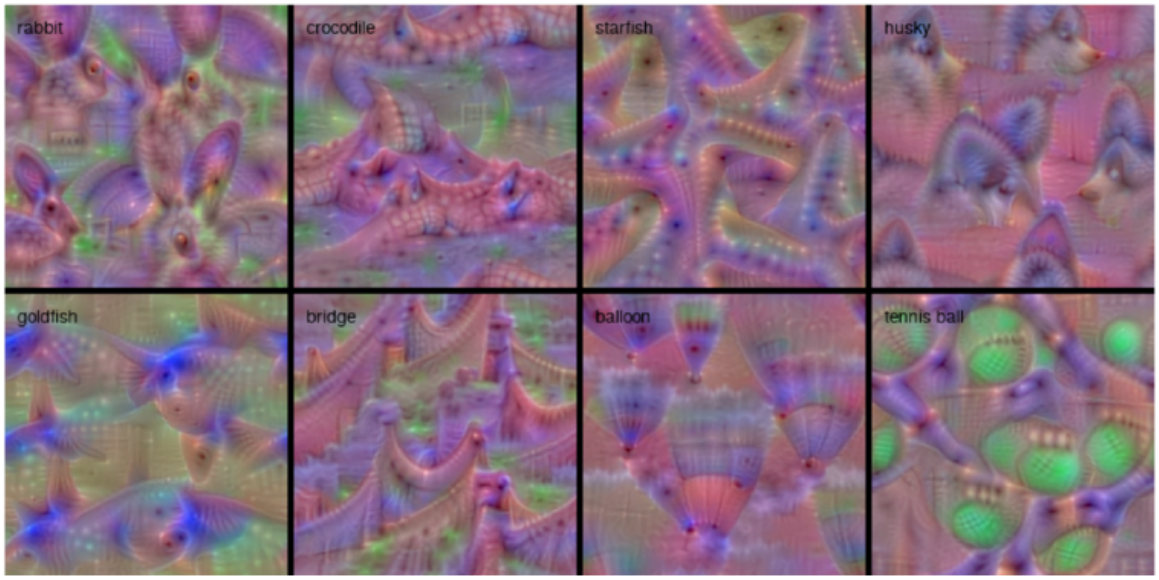
Figure 3.11: This filter represents a vertical contrast difference.



(a) Representation of the image that the first layer will maximally respond to. It contains the most basic shape or color pattern.



(b) Representation of the image that the thrid layer will maximally respond to. It contains more complex shapes in combination with some color patterns.



(c) Representation of the image that the sixth and last layer will maximally respond to. It contains the most complex shapes in combination with color. The objects are quite clearly definable.

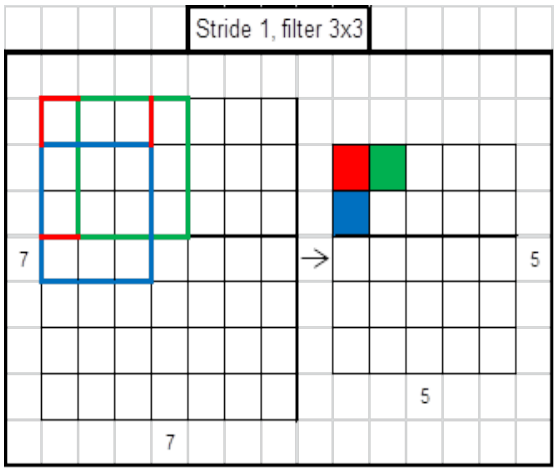
Figure 3.12: These visualizations show the images that the corresponding filter will maximally respond to.

3.5.3 Handcrafted or data generated features

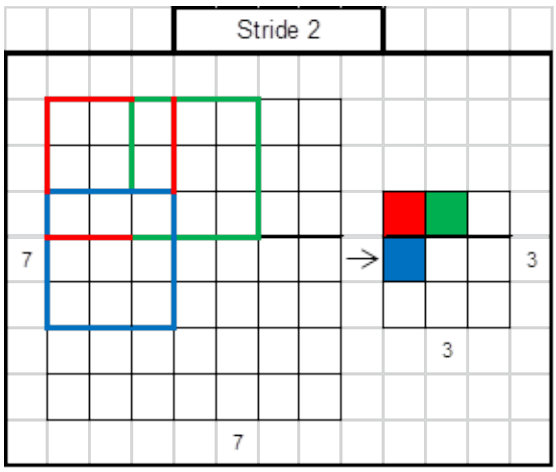
The strength of a CNN lies in the ability to adapt its filters according to training data to achieve better performance than traditional handcrafted features. This adaptability leads to CNN-extracted features to contain more adapted and complex patterns, which are useful for classification tasks. The detected patterns of CNN layers are automatically generated according based on the predefined architecture. Earlier layers tend to recognize simpler patterns and subsequent layers recognize more complex patterns. These simple and complex patterns within a network are partially visualized in Figure 3.12a, 3.12b and 3.12c. This network is trained on the ImageNet dataset [37] with the VGG16 algorithm [38].

3.5.4 Kernel sizes, stride and padding

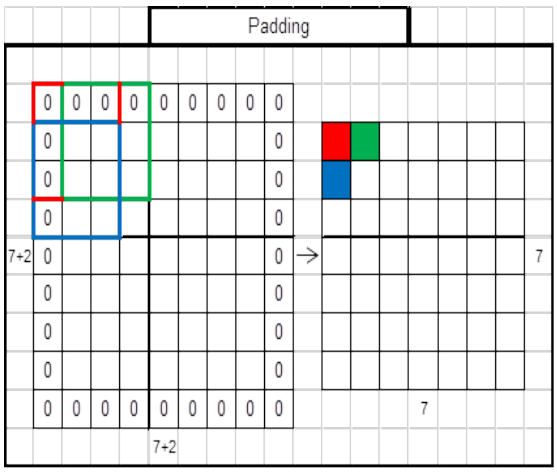
A few concepts that will be used in this work are kernel size, stride, and padding.
Kernel size is the size of the filters used in convolutional and pooling layers. Larger kernel sizes are computationally more expensive than smaller kernels and the output dimensions are far smaller, as visualized in Figure 3.13d.
Stride determines the movement of the filter in the input volume. Examples of a stride $= 1$ and stride $= 2$ filter are represented in respectively Figure 3.13a and Figure 3.13b. As shown in Figure 3.13b, a stride equal to 2 results in a reduced size of the output matrix and fewer values to process in subsequent layers. Thus, often a higher stride value is used to speed up the training phase of a network.
Padding is an edge condition and adds data to the input nodes to preserve the size. With padding the model is able to capture more information from the previous layer. An example is given in Figure 3.13c.



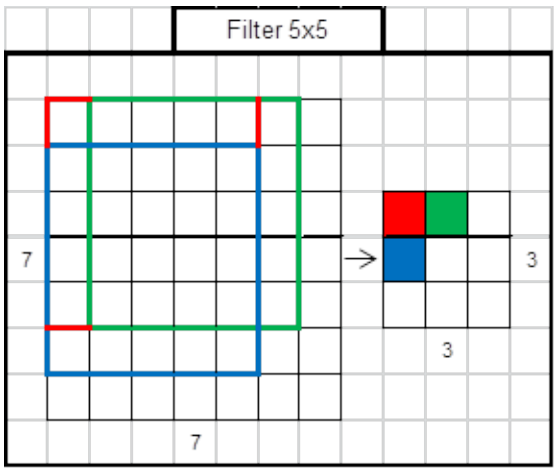
(a) This example is the most generic example, with a 3x3 filter and a stride of 1. The filter shifts one node at a time with a 3x3 filter, resulting in a 5x5 output matrix



(b) This example uses a stride of 2. The filter shifts 2 nodes at a time with a 3x3 filter, resulting in a 3x3 output matrix.



(c) This example uses padding. Padding zero values around the input matrix results in an 7x7 output matrix, thus the dimensions have been retained



(d) This example uses a 5x5 filter. This filter results in a 3x3 output matrix.

Figure 3.13: Visualization of stride, padding and kernel size.

3.5.5 Pooling layers

Convolutional layers lead to a very high computational load [56] in the form of memory usage. To reduce the memory usage and the number of parameters, pooling layers are used. Pooling layers, which often have a stride > 1, are kernels that reduce the number of nodes. An example is given in Figure 3.14, in this case it is represented as a 2×2 filter, with stride 2. A function such as the maximum value of the four values is used and returned as an output node, generating a feature map with 75% fewer nodes.

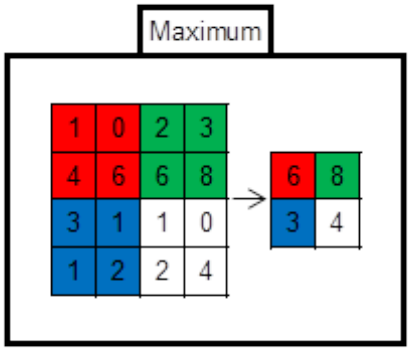


Figure 3.14: This example demonstrates how "maxpooling" works.

3.5.6 Optimizers

Gradient descent, explained in 3.4.6, is an example of an optimizer and represents one of many methods to optimize some function $f(x)$. Its task is to make the model converge to the global optimum. The difficulty with this task lies with the many local critical points to which the algorithm can converge.

In our case, the Adam algorithm [57] is used. Adam combines multiple methods to optimize its convergence into a global critical point. The algorithm uses the following methods, but is still based on the classical gradient descent:

- *Momentum*, which is calculated according to previous gradients, resulting in bigger steps if previous steps were in the same direction. In this case, the gradient is used as an acceleration for the learning rate, not as a velocity.
- *Adaptive learning rate*. The learning rate is adapted according to both the average first moment (the mean) and the second moment variance of the gradients. Due to this adaptive learning rate, tuning the learning rate hyper-parameter is less necessary.

Empirically spoken, the Adam optimization algorithm is often used, works well and converges very fast with classification tasks like the one researched in this work [55].

3.6 Recurrent Neural Networks

This chapter will discuss *recurrent neural networks* (RNN), a type of network which is designed to analyze sequences. In our case the RNN is used to classify a sequence of consecutive 2D slices representing the 3D

images.

The main difference is that all previously discussed networks are "feed forward" networks, while a recurrent neural networks "remember" parts of information. In a feed forward network the information flows from the input of the network to the output. A recurrent network feeds a part of the information back into the network (remembering) which is then used for the next sample, in this case a 2D slice. These recurrent networks are used for problems in which the sequence of images is more important than individual frames. In our data we might find evaluating single 2D slices apart from each other does not provide enough information to correctly classify the complete 3D image. Therefore, it might be useful to remember the important features of previous slices and combine those with the next slices to correctly classify the whole image. The network looks something like Figure 3.15

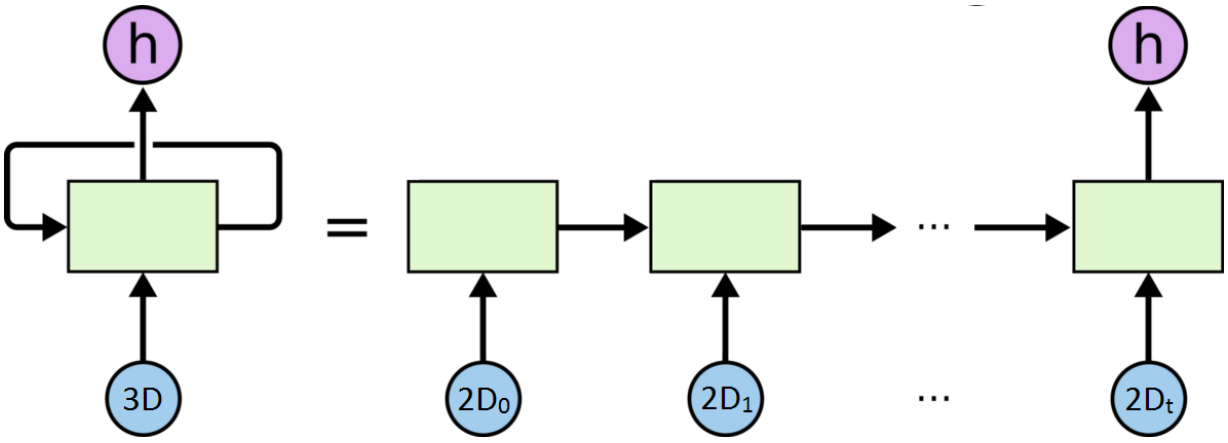


Figure 3.15: A visual representation of the RNN used for the experiments in this work. This image is based on [4].

'3D' represents the 3D input image, which flows through the RNN and outputs a classification h. When this RNN is unrolled, the first input of the network, $2D_0$, is the first 2D slice of the image, and so on. The network extracts the features from the 2D slice and these are fed into the next network. In this experiment illustrated as a new network with input $2D_1$. When the last slice, $2D_t$ is processed by the network, the output (h) is produced.

3.6.1 Backpropagation Through Recurrency

Computing gradient through a recurrent neural network is like the generalized back-propagation algorithm of section 3.4.2 calculated to the unrolled computational graph as in Figure 3.15. Recurrent networks share weights between unrolled networks, also the network has multiple input sources, thus the back-propagation algorithm will have to calculate all new shared weights and biases from $2D_t$ to $2D_0$.

3.6.2 Long Short-Term Memory

To achieve reasonable performance it is important that the final classification is based on the memorized features of important 2D slices that have been processed by the network. Unfortunately, the standard recurrent network forgets some of the memorized features of previous slices with every new slice that is processed. At the last slice of a 3D image the network has only been able to remember the features of the previous 100 slices [58]. In our dataset, the images have a z-dimension of 196 slices, thus it does not have the ability to "understand" the full image dependencies.

This is why the Long Short-Term Memory (LSTM) cell [59] is used, which is visualized in Figure 3.16. LSTM cells are an extension to regular recurrent cells, which basically extends their memory. The main idea of this cell is that by using gates[6], it is able to learn what to store in the long term memory state c, and what to store in the short term memory state h. All gates are trained through weights which translate to the networks ability to learn over time which information is important and which not.

In an LSTM cell there are three gates, these are represented in Figure 3.16. From left to right these are, the forget gate, the input gate and the output gate. These gates respectively determine:

- whether to delete the previously learned information from the long term memory state c.
- whether to let new input (x_t) in or ignore it.
- whether to let the new input and short term memory h impact the output.

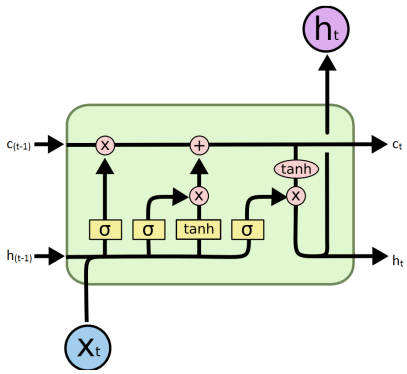


Figure 3.16: A visual representation of a LSTM cell, the three gates are indicated by a σ. This image is based on [4].

The gates used in these LSTM cells are operated by their corresponding weights. According to the weights certain information is more or less important in the calculation of the long-term memory state c and short-term memory state h. The gates use Sigmoid-functions, thus information with a weight value of zero will not be used in the calculation, while information with a weight value of one will heavily influence the output. The Sigmoid function is visualized in Figure 3.8b in Section 3.4.5.

3.7 Experiment setup

The experiments follow a certain structure. First, the images need to be preprocessed since the dimensions of the images are too large to properly train the network on, due to time and computing power constraints.

[6]Gates mean that the cell is able to decide whether or not to store (open gate) or delete (closed gate) information

This preprocessing step is explained in Section 3.7.1. An AlexNet-like architecture is designed and a neural network based on this is build and trained. This architecture is discussed in Section 3.7.2, and is the baseline for our experiments. The second architecture, which will be compared with the AlexNet-like architecture, is a combination of a CNN and RNN and is elaborated upon in Section 3.7.3.

3.7.1 Preprocessing

The images are saved in a DICOS format, which is the security derivative of DICOM, the standard for the communication and management of medical imaging information and related data [60]. The following adaptations have been made to the DICOS images:

Noise reduction: The images in the datasets are cluttered and noisy. The values of voxels[7] are in Hounsfield Units [22]. Hounsfield Unit values below 500 are materials of which firearms are not made and mostly represent biological materials. Therefore, all values below 500 are reduced to 0 which results in a less cluttered and noisy image.

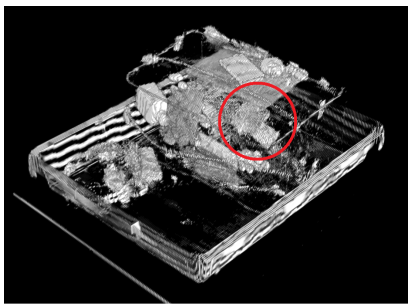


Figure 3.17: The preprocessed example, as in Figure 3.1a, the red circle is where the firearm is located.

Cubic voxel dimensions: All voxels are of size $[1, 1, 1.78]$, which is a rectangular prism. To make optimal use of the network, the size of the voxels are reshaped to a $[1, 1, 1]$ cube size. Cubic voxel dimensions are preferred since they can be rotated without causing a different representation.

Smaller image dimensions: The original dimensions of the images are too large for the network to train on with the available data. Training the network would take too long with the limited computing power that is available. Therefore, the image dimensions are reduced to $[150, 100, 196]$ voxels, which is about 25% of the original in the x- and y-direction and 43.5% in the z-direction (due to reshaping to a cubic voxel). With this preprocessing step the number of voxels in total is reduced by 97%. The method of subsampling is spline interpolation.

These preprocessing steps are applied to both the negative- and the positive-dataset so that the network is able to use the images, an example of a preprocessed image can be found in Figure 3.17.

3.7.2 AlexNet-like architecture baseline

In this work, the AlexNet architecture [5] is used with a few modifications to be able to train on 3D data. This subsection also covers the choices made for hyper-parameter optimization. AlexNet is designed to classify

[7] A voxel is the same as a pixel but has > 2 dimensions.

images like the ImageNet dataset [37] and won the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012. At the time of designing AlexNet, a single GPU was unable to load the model due to not having enough VRAM, thus its architecture was designed to be used on two GPU's. Between specific layers, there are connections leading from one parallel layer to the other to make the model use all trained filters from the previous layer. However, nowadays the GPUs are powerful enough to run AlexNet on a single GPU, these layers are displayed as the basic structure of AlexNet in Figure 3.18.

The input layer takes a colored image of 224×224 pixels. It contains five convolutional layers and two fully connected layers. The output layer has 1000 nodes, corresponding to the number of classes in the ImageNet dataset.

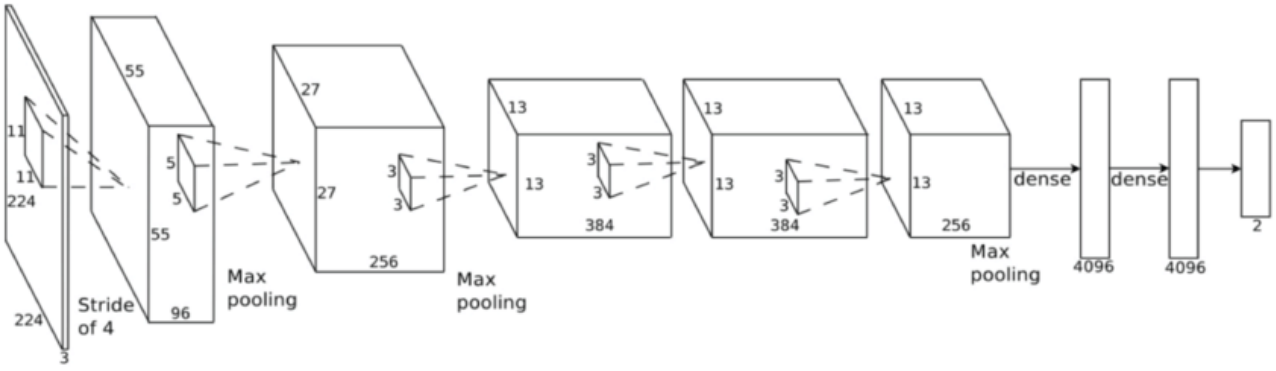


Figure 3.18: A visual representation of AlexNet [5].

The network is adapted so that it is able to process a 3D input, where the original only accepted 2D input. The convolutional layers in this network use 3D kernels, which move in all 3 dimensions, to fully utilize the information available in the CT images.

Hyperparameter optimization

The networks final architecture is presented in Table 3.2, and Figure 3.19. The input layer has a $[150, 100, 196]$ shape, the network is compiled with an Adam optimizer with a learning rate set to 10^{-4}, and the cost function is binary cross-entropy [61]. A 50% dropout is added to both fully connected layers. The training phase is in batches of 80 samples, of which there are 160 in one epoch. An early stopping callback is added to reduce training time, it keeps track of the validation accuracy and stops the algorithm when it has not improved for 50 epochs. This value is selected based on the analysis of the algorithm which ran for 300+ epochs, which can be found in Figure 3.20. This shows that the network starts overfitting from around 220 epochs (red dashed line). With these hyper-parameters, the network takes around 50 epochs to stabilize after which the validation loss is stable for over 150 epochs. Therefore, the early stopping algorithm waits 50 epochs before stopping the algorithm, after which the parameters for which the network performed best on the validation set are used.

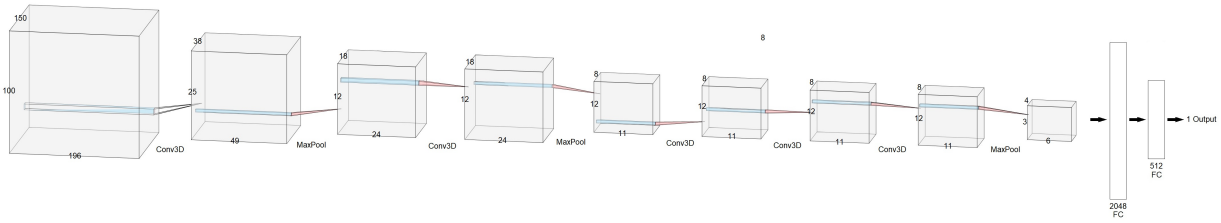


Figure 3.19: A visualization of the 3D AlexNet-like architecture used in our first experiment. The visualization is simplified, due to the 4D nature of the architecture, into 3D space.

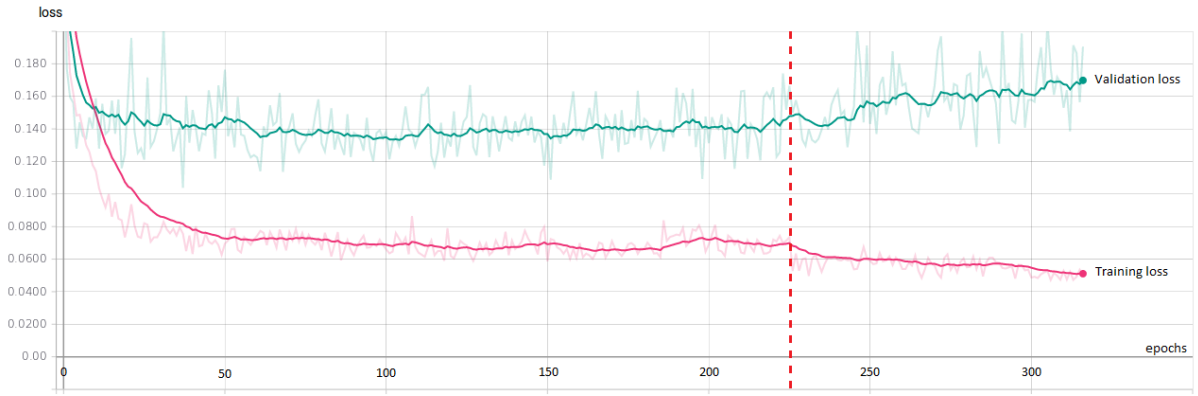


Figure 3.20: A visualization of the training and validation losses. The training and validation dataset are respectively 80% and 10%

Table 3.2: The complete AlexNet-like network architecture.

Name	Units	Kernel	Stride	Activation
3D Convolutional Layer #1	96	[11,11,11]	[4,4,4]	ReLU
3D Max Pool Layer #1	—	[3,3,3]	[2,2,2]	ReLU
3D Convolutional Layer #2	256	[3,3,3]	[1,1,1]	ReLU
3D Max Pool Layer #2	—	[3,3,3]	[2,2,2]	ReLU
3D Convolutional Layer #3	384	[3,3,3]	[1,1,1]	ReLU
3D Convolutional Layer #4	384	[3,3,3]	[1,1,1]	ReLU
3D Convolutional Layer #5	256	[3,3,3]	[1,1,1]	ReLU
3D Max Pool Layer #3	—	[3,3,3]	[2,2,2]	ReLU
Fully Connected Layer #1	2048	—	—	ReLU
Fully Connected Layer #2	512	—	—	ReLU
Output Layer	1	—	—	Sigmoid

3.7.3 A CNN/RNN architecture

The second experiment uses a different architecture for the classification of 3D images. This architecture is based on a combination of both CNN and RNN properties. The CNN extracts features from consecutive 2D slices of the 3D CT image, then a RNN processes the 2D slices resulting in one classification. The CNN part of the architecture are five 2D AlexNet [5] layers which extract the features of the 2D slices. The RNN part of the architecture utilizes two LSTM layers followed by a dense layer and the output layer.

Hyperparameter optimization

The CNN/RNN networks final architecture is presented in Table 3.3, and Figure 3.21. The networks is compiled with an Adam optimizer with a learning rate set to 10^{-4}, and the cost function is binary cross-entropy [61]. A 50% dropout is added to both LSTM layers and the dense layer. The training phase is in batches of 20 samples, of which there are 640 in one epoch. The early stopping algorithm is also identical, by waiting 50 epochs for improvement. This value is again picked based on the analysis of the algorithm which ran for 300 epochs, which can be found in Figure 3.22. This time there is no clear overfitting within the 300 epochs, but the network stabilizes after around 50 epochs and is stable for the other 250.

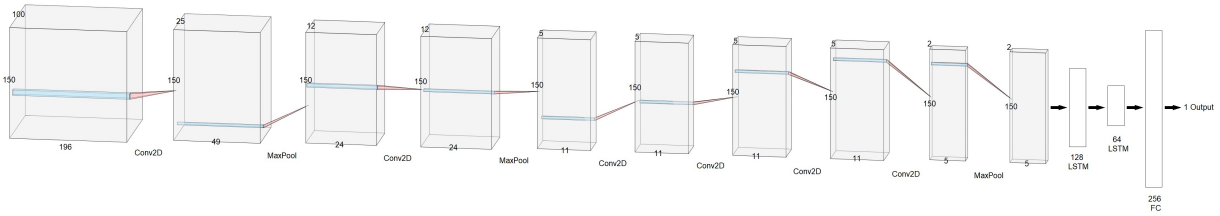


Figure 3.21: A visualization of the CNN/RNN architecture used in our second experiment.

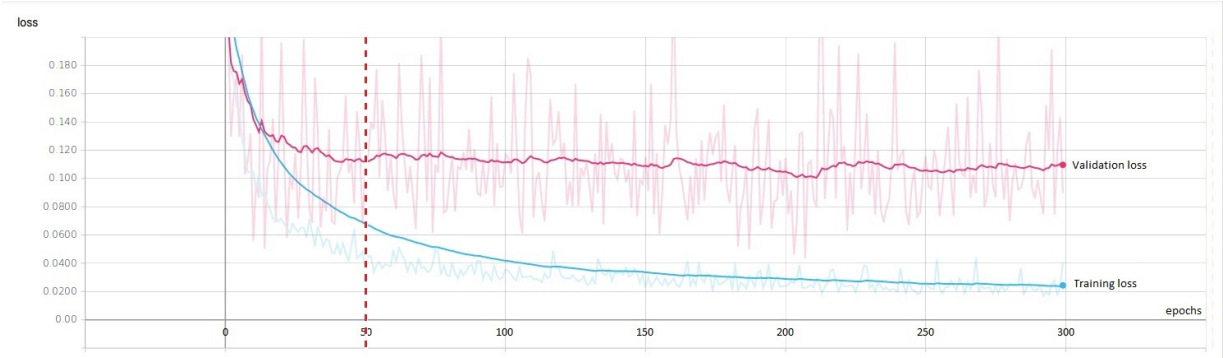


Figure 3.22: A visualization of the training and validation losses. The training and validation dataset are respectively 80% and 10%

Table 3.3: The complete CNN/RNN network architecture.

Name	Filters	Kernel	Stride	Activation
Convolutional Layer #1	96	[11,11]	[4,4]	ReLU
Max pool Layer #1	—	[3,3]	[2,2]	ReLU
Convolutional Layer #2	256	[5,5]	[1,1]	ReLU
Max pool Layer #2	—	[3,3]	[2,2]	ReLU
Convolutional Layer #3	384	[3,3]	[1,1]	ReLU
Convolutional Layer #4	384	[3,3]	[1,1]	ReLU
Convolutional Layer #5	256	[3,3]	[1,1]	ReLU
Max pool Layer #3	—	[3,3]	[2,2]	ReLU
LSTM layer #1	128	—	—	tanh
LSTM layer #2	64	—	—	tanh
Fully Connected Layer #1	256	—	—	ReLU
Output Layer	1	—	—	Sigmoid

3.8 Result validation

Result validation is necessary for any classification algorithm to detect and prevent over- and underfitting. It is important to know the certainty of which the classification result of the network is correct or incorrect. The risk assessment of using an algorithm instead of a security officer can be based on these values. To tackle this problem, and measure the networks quality, extensive testing is necessary. This section suggests multiple metrics to validate the results of the network.

To calculate the metrics used in this section, a confusion matrix is used, which is visualized in Table 3.4. There are four possible classification options based on whether the image contains a firearm (y), the algorithm classified it as containing a firearm ($\hat{y}$), the image does not contain a firearm($\neg y$), or the algorithm classified the images as not containing a firearm ($\neg \hat{y}$).

Table 3.4: A confusion matrix with four possible classification options.

	$\hat{y}$	$\neg \hat{y}$
y	True Positive (TP)	False Negative (FN)
$\neg y$	False Positive (FP)	True Negative (FN)

The output of the network is a number between zero and one. Whether the network classifies an image as positive or negative is based on a certain threshold. This threshold, θ, states that for every output value less than or equal to θ it evaluates that image as negative. For every output value greater than θ it evaluates the image as positive. A higher threshold results in a higher number of false negatives and a lower number of false positives, and a lower threshold has the opposite effect. In summary, by adjusting the threshold a trade-off between false negatives and false positives can be made.

The classes in our dataset are not evenly distributed, there are approximately 91% negative images and 9% positive images. When using the accuracy[8] metric, 91% accuracy would be easily achieved by classifying

[8]accuracy is the ratio of correctly classified images out of all images.

every image as negative, which seems to be great, but is not. Therefore, other metrics are used, like *sensitivity*, *specificity*, and *precision*.

In the next section, the ROC curve is discussed with the associated sensitivity and specificity metrics. Then, precision and recall are discussed with its corresponding PR curve in Section 3.8.2. Finally, k-fold cross-validation and its use is explained in Section 3.8.3.

3.8.1 ROC curve

The Receiver Operating Characteristics (ROC) curve is created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR), at various settings for the threshold θ. TPR is also called Sensitivity, and FPR is equal to 1 - Specificity.

The *sensitivity* metric is the fraction of images that have been correctly detected as containing a potential harmful item against the total amount of images with a potential harmful items. Equation (3.12) corresponds with the sensitivity metric.

$$Sensitivity = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \tag{3.12}$$

The *specificity* metric is the fraction of images that have been correctly detected as not containing any potential harmful item against the total number of safe images. Equation (3.13) corresponds with the specificity metric.

$$Specificity = \frac{\text{True Negative}}{\text{True Negative + False Positive}} \tag{3.13}$$

The specificity and sensitivity metrics are then plotted on the x- and y-axes of the ROC-curve. A convex hull is plotted through calculated points of specificity and sensitivity values with corresponding thresholds. A random guess would return a point along the dotted diagonal line, as in Figure 3.23. The area under the ROC curve (AUROC) is often used to measure the performance of a classifier independent to the threshold θ.

Unfortunately, due to the ratio of positive samples and negative samples in the dataset the ROC curve is not suitable to represent the performance of the model. The specificity is dominated by the number of true negative samples over the number of false positive samples. This leads to a FPR value of approximately 1 with most θ values. As the FPR is one of the ROC curve dimensions, this leads to the ROC only having limited usefulness.

3.8.2 Precision and recall curve

To take the unbalanced ratio of positive and negative samples into account, the *recall* metric is used in combination with the *precision* metric. Recall (or detection rate) is equal to the sensitivity metric, as in Equation (3.12). The *precision* metric is the fraction of images that have been correctly detected as containing a potentially harmful item out of the total amount of images *classified* as containing a potentially harmful item. Equation (3.14) defines the precision metric.

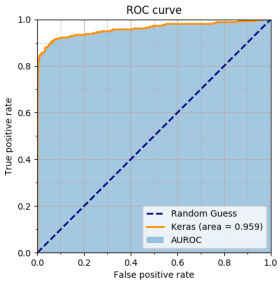


Figure 3.23: A visualization of a ROC curve.

$$Precision = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \tag{3.14}$$

The Precision-Recall (PR) curve is created by plotting the recall metric against the precision at various threshold settings. An optimal threshold can be selected based on the cost of false negatives against the cost of false positives. The cost can, for example, be the hourly rate of a security officer for opening falsely positive labeled baggage, this is further elaborated upon in Chapter 6. The PR curve also has an Area Under Curve called AUPR or sometimes AP (Average Precision), which is a measure independent to the threshold θ.

False alarm over the total population

Figure 3.24 is an abstract visual representation of a cabin baggage screening belt. This figure contains three belts on which baggage is transferred. The drop-off area is where the passengers drop their baggage to be scanned by the CT machine. The Safe labeled baggage (green), which is the belt where passengers can immediately grab their belongings since there were no potentially harmful items detected. The Unsafe labeled baggage (red), which is the belt where passengers' baggage is searched through, due to the detection of potentially harmful items.

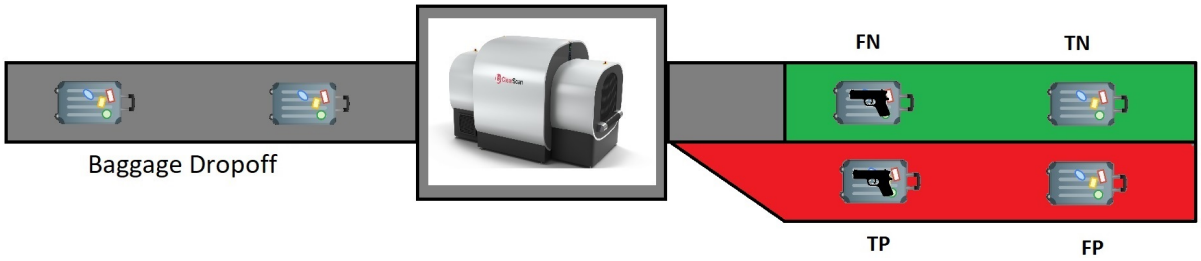


Figure 3.24: An abstract visualization of the cabin baggage screening process. The red lane is baggage predicted as unsafe, while the green lane is baggage predicted as safe. Image of CT scanner provided by L3 Security and Detection Systems

The classic false alarm rate (FAR) is calculated as Equation (3.15). The false alarm rate calculates the ratio between baggage which is flagged as containing a firearm and the baggage which actually contains a firearm.

This results in a percentage of bags which are opened by a security officer, while unnecessary. A higher FAR thus results in higher baggage screening times.

On the other hand, Schiphol uses a "false alarm rate over the total population" (FATP) metric. The main reason for this is the real-life TN value is unknown, since the FN value is unknown. Only the total of the FN and TN is known, thus FATP uses the ratio of FP over all baggage. Also, this metric is the easy derivation of additional costs for security officers opening baggage, which translates into work load, and the speed of screening baggage. The false alarm of the total population metric is shown in Equation (3.16). The currently used minimal requirements for at CT machinery dutch airports is a minimum detection rate of CONF% with a FATP maximum of CONF%.

$$FAR = \frac{FP}{FP + TN} \tag{3.15}$$

$$FATP = \frac{FP}{FP + TN + TP + FN} \tag{3.16}$$

3.8.3 k-fold cross-validation

Cross-validation is used to accurately test the model's performance on a fold which was not used in the training phase. In our experiments the dataset is split in 10 folds (subsets), of these folds, 8 are used for training, 1 for validation, and 1 for testing purposes. The results from these experiments are used to detect overfitting problems. It will also give insights in how the network will behave to an unknown dataset.

When dividing the dataset in folds, there is a chance the network, which is trained on the training folds, is great at classifying the test fold due to the random distribution. While another random distribution would have resulted in lesser results. To reduce this variability in the results of the network, the experiment is repeated k times, in this case 10. These 10 folds are distributed over 10 iterations according to the shuffle scheme in Figure 3.5.

Table 3.5: A visual representation of a 10 Fold cross-validation as conducted in this work.

	Full Dataset, 28,969 images									
Iteration 1	Train	Train	Train	Train	Train	Train	Train	Train	**Validate**	**Test**
Iteration 2	Train	Train	Train	Train	Train	Train	Train	**Validate**	**Test**	Train
Iteration 3	Train	Train	Train	Train	Train	Train	**Validate**	**Test**	Train	Train
Iteration 4	Train	Train	Train	Train	Train	**Validate**	**Test**	Train	Train	Train
Iteration 5	Train	Train	Train	Train	**Validate**	**Test**	Train	Train	Train	Train
Iteration 6	Train	Train	Train	**Validate**	**Test**	Train	Train	Train	Train	Train
Iteration 7	Train	Train	**Validate**	**Test**	Train	Train	Train	Train	Train	Train
Iteration 8	Train	**Validate**	**Test**	Train	Train	Train	Train	Train	Train	Train
Iteration 9	**Validate**	**Test**	Train	Train	Train	Train	Train	Train	Train	Train
Iteration 10	**Test**	Train	Train	Train	Train	Train	Train	Train	Train	**Validate**

Finally, due to the random initialization of weights and biases there is a chance one network is able to converge in a near global optimum, while another random initialization might be unable to get anywhere near. To reduce this variability, the 10-fold cross-validation is repeated 3 times, with different random initialization. The test results of these runs and their iterations are combined to give an estimate of the model's predictive performance.

Static seed value

The last step towards proper result validation is a static seed value. This ensures identical weight and bias initialization in the same run with different folds. It also ensures identical dataset distribution of the 10 folds within a run. in between runs, there are different weights, biases and fold distributions initialized. The fold distributions is identical between the AlexNet-like experiment and the CNN/RNN experiment.

3.8.4 Implementation details

The 3D AlexNet-like network was trained with a batch size of 80 and the CNN-RNN network with a batch size of 20. After each epoch, the validation loss was calculated using the validation fold. The network stops training when the validation loss has not decreased for 50 consecutive epochs.

Chapter 4

Results

This Chapter will discuss the results achieved by the AlexNet-like algorithm, and the CNN/RNN algorithm in respectively Section 4.1, and Section 4.2. The training process was, for both algorithms, cross-validated with 10 folds, in which the weights and biases were initiated the identical. It is ran 3 consecutive times with different initiated weights and biases between runs. Both algorithms were run on the hardware and software described in Section 3.7.

4.1 Results of 3D AlexNet-like algorithm

The results of the experiment discussed in Section 3.7.2 are presented in Table 4.1. These results were calculated after the network was trained for an average of 58 epochs. The training phase ran for 50 additional epochs due to the stopping criteria mentioned in Section 3.8.4. This process took about 3 weeks to complete, with one model taking up to 24-26 hours to train.

The results are displayed in Table 4.1 and Figure 4.1. This table shows the corresponding recall, precision, false alarm rate (FAR), false alarm rate over the total population (FATP), and accuracy values for a certain threshold which is based on the recall values of 100%, 99%, 95%, 90%, and 80%.

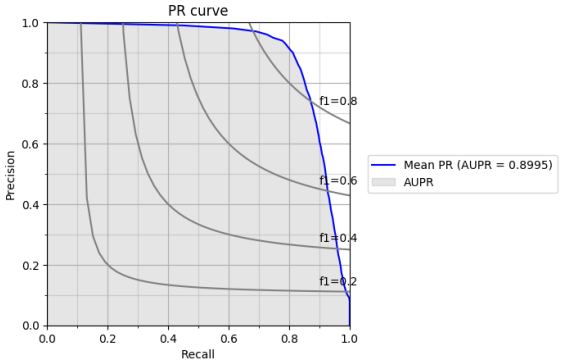
Table 4.1: The results from the AlexNet-like network, a 95% confidence interval is shown in parentheses.

Recall	**Precision**		**FAR**		**FATP**		**Accuracy**		**TP**	**FN**	**FP**	**TN**
100.0	8.9	(± 0.0)	99.9	(± 0.3)	91.0	(± 0.3)	9.0	(± 0.3)	7,695	0	78,920	85
99.0	10.3	(± 8.2)	84.3	(± 29.1)	76.5	(± 26.6)	23.1	(± 26.4)	7,619	76	66,340	12,365
95.0	20.8	(± 19.5)	35.3	(± 35.1)	32.0	(± 32.0)	67.4	(± 31.7)	7,311	384	27,760	50,945
90.0	55.2	(± 13.4)	7.1	(± 5.3)	6.5	(± 4.9)	92.6	(± 4.6)	6,926	769	5,611	73,094
80.0	90.2	(± 4.3)	0.9	(± 0.4)	0.8	(± 0.4)	97.4	(± 0.3)	6,156	1,539	672	78,033

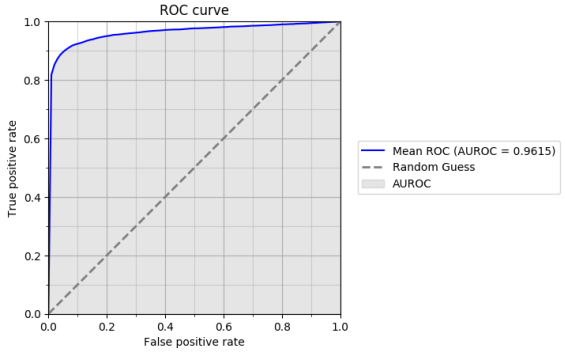
These values are averaged over 3 runs of 10 folds and are based on a single threshold, which is selected based

on the first recall value slightly above the target recall value. The baseline performance of an operator assist algorithm algorithm is set to an CONF% recall (detection rate/sensitivity). As visualized in Table 4.1, we have also tested 90% recall and higher. However, a further increase in the recall value above 90% results in low precision and high false alarm rate values. We found that this does not add any value to a potential operator assist system to be used at Schiphol airport.

A complete overview of possible recall and corresponding precision values can be derived from the PR-curve displayed in Figure 4.1a. Although the ROC curve is not suitable to represent the performance of the model, as mentioned in Section 3.8.1, it is still displayed in Figure 4.1b as a visualization of the true and corresponding false positive rate.



(a) Visualization of the 3-run combined PR curve generated from the AlexNet-like networks, with an area under the curve score of 0.8995



(b) Visualization of the 3-run combined ROC curve generated from the AlexNet-like networks, with an area under the curve score of 0.9615

Figure 4.1: A visualization of both the ROC and PR curve of the AlexNet-like networks.

4.2 Results of combined CNN/RNN algorithm

The results of the experiment discussed in Section 3.7.3 are presented in Table 4.2 and Figure 4.2. For this network, the training phase ran for an additional 50 epochs, based on the stopping criteria in Section 3.7.3. This process took about 2.5 weeks, with one model taking up to 18-20 hours to train.

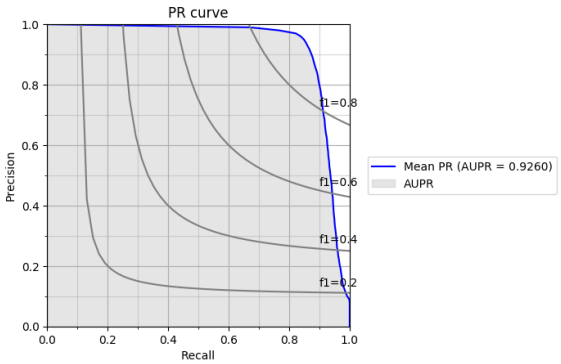
Table 4.2 shows the corresponding recall, precision, false alarm rate (FAR), and accuracy values for a specific threshold which is based on the recall values of 100%, 99%, 95%, 90%, and 80%.

Table 4.2: The results from the CNN/RNN network, a 95% confidence interval is shown in parentheses.

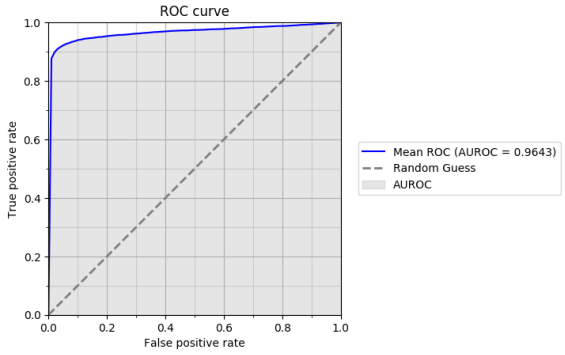
Recall	Precision		FAR		FATP		Accuracy		TP	FN	FP	TN
100.0	8.9	(± 0.0)	99.9	(± 0.1)	91.0	(± 0.1)	9.0	(± 0.1)	7,695	0	78,665	40
99.0	10.4	(± 4.2)	83.8	(± 21.1)	76.4	(± 19.3)	23.5	(± 19.2)	7,619	76	65,991	12,714
95.0	30.2	(± 16.1)	21.4	(± 20.9)	19.5	(± 19.0)	80.0	(± 18.9)	7,312	383	16,865	61,840
90.0	76.5	(± 10.7)	2.7	(± 2.1)	2.5	(± 2.0)	96.7	(± 1.8)	6,926	769	2,123	76,582
80.0	97.6	(± 2.0)	0.2	(± 0.2)	0.2	(± 0.2)	98.0	(± 0.4)	6,156	1,539	152	78,553

These values are averaged over 3 runs of 10 fold cross-validation and are based on a single threshold, which is selected based on the lowest recall value above the minimum. Again, the baseline performance is CONF% but in the case of the CNN/RNN algorithm, a 90% recall results in a 2.7% false alarm rate. Any recall value above 90% results in low precision and high false alarm rate values.

An overview of possible recall and corresponding precision values can be derived from the PR-curve displayed in Figure 4.2a. The ROC curve is displayed in Figure 4.2b as a visualization of the true positive and corresponding false positive rate.



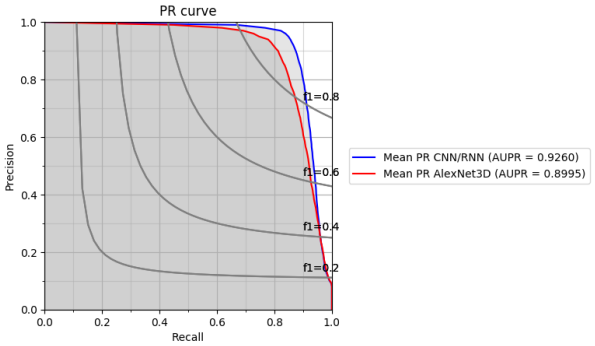
(a) Visualization of the 3-run combined PR curve generated from the CNN/RNN networks, with an area under the curve score of 0.9260



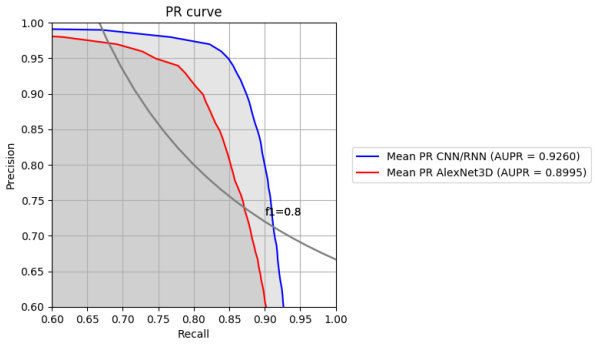
(b) Visualization of the 3-run combined ROC curve generated from the CNN/RNN networks, with an area under the curve score of 0.9643

Figure 4.2: A visualization of both the ROC and PR curve of the CNN/RNN networks.

To compare the ROC and PR curves of both networks, the figures are combined into Figure 4.4 and Figure 4.3. As we can see in close up Figure 4.3b and Figure 4.4b, there are no recall values where the AlexNet3D network performs better than the CNN/RNN network.

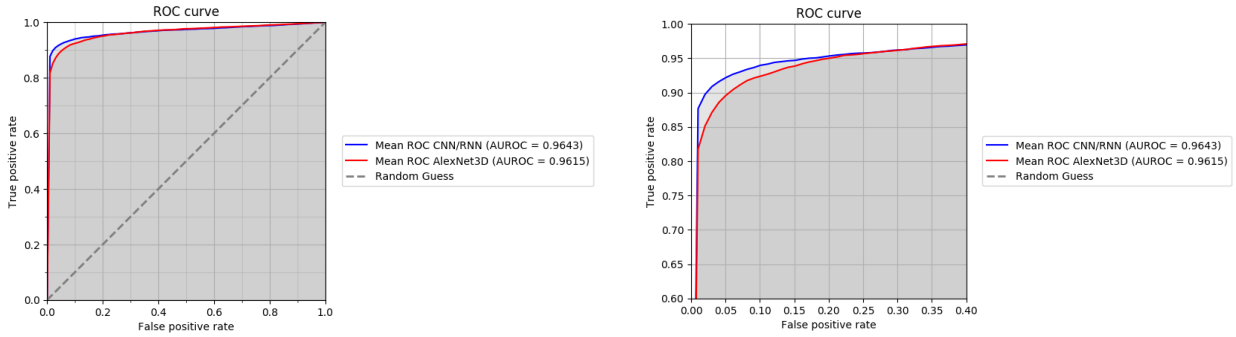


(a) A combined PR curve of both the AlexNet3D and the CNN/RNN networks.



(b) Figure 4.3a but zoomed in on precision and recall values above 0.6

Figure 4.3: A combined PR curve of both algorithms and a zoomed in version.



(a) A combined ROC curve of both the AlexNet3D and the CNN/RNN networks.

(b) Figure 4.4a but zoomed in on precision and recall values above 0.6

Figure 4.4: A combined ROC curve of both algorithms and a zoomed in version.

Chapter 5

Adverserial Learning

Adverserial learning refers to a technique of generating adverserial input with the intent to mislead a targeted model [62]. Carefully crafted perturbations are generated for both targeted and non-targeted attacks. A targeted attack is when the attacker misleads the classifier to misclassify as a specific target class. A non-targeted attack is where the attacker aims for misclassification. Liu *et al.* [63] states that, in case of a non-targeted attack, adversarial examples generated for one model, can be transferred to be used to another model. This research also states the targeted attacks almost never transfer when used on a different model.

Since this research is conducted for airport security, the possibility to mislead the system is problematic. The available adversarial learning methods and the possible impact on our model are discussed in Section 5.1. Then, to prevent malevolent people from abusing the technological advances of using neural networks for automated cabin baggage check, some countermeasures are mentioned in Section 5.2.

5.1 Available methods

This Section reviews the three main methods of adversarial learning and the corresponding influence on the security checkpoint algorithms:

- Digital injection of adversarial network generated pixel-values, Section 5.1.1
- Physical injection of adversarial network generated "patches", Section 5.1.2
- Physical injection of adversarial network generated 3D printed items, Section 5.1.3

5.1.1 Digital injection of adversarial network generated pixel-values

Starting with the digital injection of pixel-values, adverserial learning is often used in cases where the input of networks are adjustable. For this method the user is able to define the input of the network and can change pixel values. For example Figure 5.1, for human vision there is a clear image of a panda, while an algorithm misclassifies it as a gibbon by the added (seemingly random) noise. However, this "noise" is carefully crafted by a second neural network trained to mislead the first neural network. For this method the output of the first network must be readable as input for the second network to be able to train the second network until it is able to mislead the first network.

On the other hand, Ilyas *et al.* [64] examined adversarial learning using query- or label-only access to the model instead of having access to its entire output, calling this "black-box adversarial learning". Query access is where the model replies the confidence values of a certain image, label access is where only the label of an image is replied. In our case, you could state that any passenger who is travelling by plane, has label access to the model and is thus able to use this black-box method to possibly mislead the system. However, the median number of queries needed with label-only access is over 2.5 million, which makes this method impossible to be used on airports checkpoints.

This thesis is based on a closed system which generates the 3D images of real life baggage as the input of the networks used for experiments. This system makes it unlikely for the images to be digitally injected with such noise and is therefore unlikely to disturb the system. However, there will always be a chance of malevolent people to obtain a CT machine, the algorithm, or both, which may lead to a compromised system.

5.1.2 Physical injection of adversarial network generated "patches"

Another method of adverserial learning is the design of a certain "patch" which is added to the image space of the object to mislead the classifier [7]. These patches are universal, can be used to attack any scene and can cause a classifier to output a target class. These patches can be printed, added to a scene and presented to the classifier. Even when patches are small, they can cause a classifier to ignore other items and report a chosen target. For example, in Figure 5.2, a physical patch is added to the scene which causes the classifier to detect a toaster instead of a banana.

Based on the nature of our data the possible addition of patches pose a serious threat towards the implementation of neural networks at security checkpoints. Due to this methods ability to be physically added to a bag or suitcase, it might be able to disrupt the algorithm. On the other hand, the CT machines make a 3D representation of baggage in which, due to the use of x-rays, information is captured of the substances and materials of items in there, and not the color. Since a neural network uses all information available it com-

bines the shape and material composition of an item to classify it as potentially harmful or not. That would mean that the network should not be disrupted by a patch as in Figure 5.2. Unfortunately, there has not been any research conducted in this field. However, if it is possible to produces patches of different materials and 3D shapes, there is a high probability malevolent people will take advantage of this method.

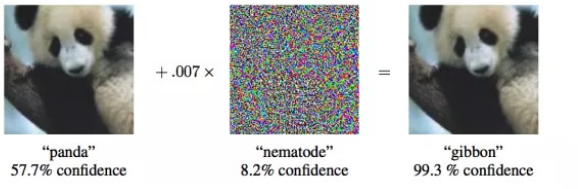


Figure 5.1: This classification network is misleaded by adversarial learning and now classifies an image of a Panda as a Gibbon [6]

Figure 5.2: This classification network is by a patch and now classifies an image of a banana as a toaster [7]

5.1.3 Physical injection of adversarial network generated 3D printed items

A third method is the addition of perturbations to items. Eykholt *et al.* [8] used perturbations in the form of only black and white stickers on traffic signs to cause misclassification for the automated driving use case. The aim of the experiments is for objects to be misclassified, while not looking suspicious to the human eye. Therefore, more tests have been conducted with subtle posters or graffiti-like perturbations. All these perturbations can be found in Figure 5.3. Athalye *et al.* [9] took it one step further and designed a complete perturbed item using a 3D printer. In this case, as in Figure 5.4, a 3D printed turtle is misclassified as a rifle.

This method of adversarial attacks brings a high risk to the table. 3D printers are widely available for consumers and therefore these 3D printable items which are misclassified by the algorithm are a big threat for the development and deployment of such algorithms. Luckily, the CT machinery returns an image containing information about the materials used in items, which may help distinguish real rifles from 3D printed turtles.

Figure 5.3: Three types of perturbations, all intended to make the classifier misclassify the sign as speed limit 45 mph [8]

Figure 5.4: The 3D printed turtle which is misclassified as a rifle [9]

5.2 Countermeasures

This section will discuss some of the available countermeasures against adversarial attacks. Papernot *et al.* [65] composed the following adversarial countermeasure requirements:

- Low impact on the architecture: any modification to the architecture needs extensive testing and analysis of its behaviour.
- Maintain accuracy: countermeasures should not significantly decrease the classification performance.
- Maintain speed of network: countermeasures should not impact the running time of the classifier, if this results in a bottleneck at the classification stage.

The upcoming three section will discuss available countermeasures to adversarial learning. Adversarial training will be discussed in Section 5.2.1, feature squeezing in Section 5.2.2, and Section 5.2.3 will consider defensive distillation.

5.2.1 Adversarial Training

One of the available countermeasures is adversarial training, both Goodfellow *et al.* [6] and Huang *et al.* [66] added adversarial examples to the training dataset. The objective of this method is to increase the models robustness by training the network on perturbed images that would otherwise mislead it. From the generated adversarial examples we might be able to understand what shapes and materials might lead to future innovative attacks.

5.2.2 Feature Squeezing

Another available countermeasure is feature squeezing, which according to Xu *et al.* [67] can be used to harden models on image classification. The idea behind this countermeasure is that as the complexity of data is reduced, the adversarial perturbations will disappear because of low sensitivity. Feature squeezing consists of two separate methods:

1. The use of a smoothing filter
2. The reduction of color depth on pixel level

The first method is already implemented in the algorithms discussed in this thesis. It is implemented in the preprocessing step, in which the dimensions of the images are reduced by smoothing multiple inputs into a single value using interpolation. This reduces noise and makes the model robust against adversarial attacks.

The second method is partially implemented, as discussed in Section 3.7.1 all values below 500 are reduced to 0. As far as the adversarial perturbation ranges values smaller than 500, it will not be able to disrupt the algorithm. However, when the material has a Hounsfield Unit value higher than 500, it will likely influence the system. The risk with this method comes with the production of potentially harmful items from materials which will have a Hounsfield Unit below 500, becoming invisible for the network.

5.2.3 Defensive distillation

Defensive distillation is a countermeasure which applies a second "distilled" network. Distillation is a training procedure whereby a second model is trained to predict the output probabilities of another model that was trained earlier [68]. It attempts to smooth the model's decisions that could be exploited [65]. This works by training the first model with "hard" labels (100% probability an image is positive rather than negative) and the second one with "soft" labels (95% probability an image is positive rather than negative). The second, distilled, model is more robust to attacks, should be used in combination with the first model [68].

One of the main advantages of the distillation countermeasure is that it is adaptable to unknown threats, which will always be attracted to the airport security industry. It is requires less human intervention, and does not require a continuous feed of all known vulnerabilities and attacks. On the other hand, the biggest disadvantage is that with enough computing power and fine-tuning on the attackers part, both models can be reverse-engineered to discover fundamental exploits. This is due to the fact that the second model is still bound by the general rules of the first model [68].

Chapter 6

Discussion

This chapter discusses the result interpretation in Section 6.1, the impact on the cost of security at Schiphol airport in Section 6.2, the limitations of the experiments in Section 6.3, and the possible future work which has emerged while writing this work in 6.4.

6.1 Result interpretation

The interpretation of the results in Chapter 4 will be discussed, and a comparison will be made to the work of Akcay *et al.* [1]. The interpretations are divided in two subsections, which are based on the security implications mentioned in Section 1.2; *"security effectiveness"*, and *"operational impact"*.

6.1.1 Comparison to previous work

In Section 2.2.2, the work of Akcay *et al.* [1] is mentioned as the latest and most successful research on the detection topic for 2D x-ray images. When comparing their binary classification results to the ones found in this study, they outperform this study with a much higher detection rate (99.56% against our 90%) in combination with a much lower false alarm rate (1.07% against our 2,5%). However their research is conducted on a different dataset and is thus not directly comparable.

Their work compared their binary classification performance to that of conventional Bag of Visual Words methods, of which the highest scoring method has a detection rate of 79.2% with a corresponding 3.2% false alarm rate. These results are outperformed by both architectures researched in this study. The false alarm rates are 0.9% (AlexNet-like), and 0.2% (CNN/RNN) for a 80% detection rate. Unfortunately, no AUPR or AUROC values are reported in their work, and could thus not be compared to the ones in this work.

6.1.2 Security effectiveness

From Table 3.2, Table 4.2, and the minimum viable product requirements mentioned in Section 3.8.2, we can state that both algorithms described in this work, outperform the minimum standard for automated detection software, which is an CONF% detection rate with a maximum FATP of CONF%.

The AlexNet-like architecture, has a 0.8% FATP for an 80% detection rate. However, when the detection rate is increased to 90% the FATP will rise so that it will impact Schiphol's security checkpoint logistics.

The CNN/RNN architecture has a 0.2% FATP for an 80% detection rate. A 90% detection rate would, with this method, still results in a FATP lower than the minimum standard for automated detection software (CONF%). This will likely not impact the logistics at one of Schiphol's security checkpoints.

Based on the results in both Table 3.2 and Table 4.2, the CNN/RNN architecture seems to be more promising than the AlexNet-like architecture. This is based on the fact that the precision, FAR, FATP, and accuracy values are all better for the tested recall values when using the CNN/RNN architecture. This can also be found in the corresponding PR curves. The AUPR of the CNN/RNN architecture is with a value of 0.9260 almost 3% greater than the AUPR of the AlexNet-like architecture, which has a value of 0.8995. From Figure 4.3 it seems like the CNN/RNN network is better for all recall values over the AlexNet3D network. However, both algorithms can still be optimized, after which the AlexNet-like architecture might outperform the CNN/RNN architecture.

If the CNN/RNN architecture based algorithm will be developed into a fully functioning automated baggage classification system, we could for example:

- reduce the FATP at an 80% detection rate by at least 1.8%, resulting in fewer bags that need to be opened due to false alarms,
- or keep a similar FATP but raise the detection rate to 90%, resulting in less undetected potentially harmful items without impacting logistics.

Manual inspection of misclassified images

When manually inspecting the hard-to-classify images, a high number of the images checked contained firearms in shapes that a layperson might also not recognize. These items deviate much from the other generalized firearms, since they are concealed in non-firearm shapes, which leads to them not being classified as harmful. For example, in Figure 6.1 there are both a cell phone and a foldable firearm present in the bag and this image is misclassified as safe. This might be due to the low number of examples of this gun (0.015% or 43 images).

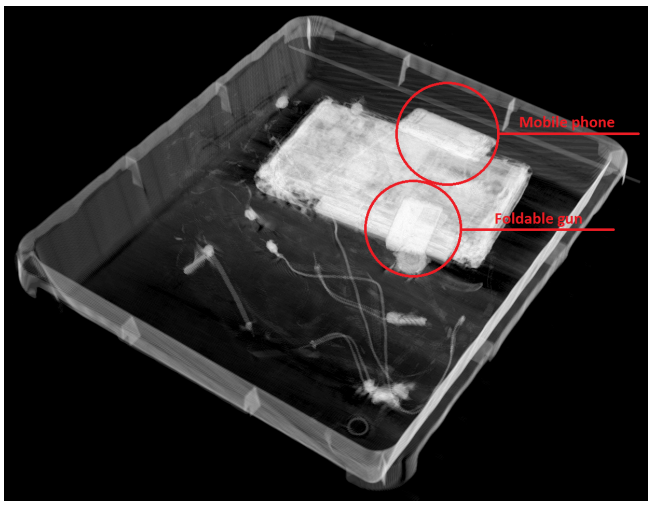


Figure 6.1: An example image of a bag which is misclassified. It shows the slight difference between mobile phones and foldable firearms

6.2 Operational impact at Schiphol

Schiphol has operational expenses for the security officer at security checkpoints. These officers make sure the cabin baggage is checked for any prohibited items. These expenses might get influenced by the introduction of automated monitoring software, possibly speeding up the process and automating certain tasks in the process.

6.2.1 Costs savings per double lane per year at a security checkpoint

Schiphol has five security checkpoints throughout the airport, which ensures a fast and smooth process flow for the passengers. Every security lane at Schiphol (Figure 6.2) shares security officer resources with the parallel lane. This is possible due to the combined working area behind the conveyors. Two of these parallel lanes are called a dual lane.

Figure 6.2: A visual of the double security checkpoint lane.

Each of these dual security lanes is occupied by 10 security officers, who all have their particular role. Of these 10 officers, 2 are monitoring the CT images of cabin baggage and deciding whether they need to perform a more thorough check. In the future, Schiphol will investigate the possibility to add another security officer on the monitoring task, to speed up the decision time on the CT images. This adds up to a total of 11 security officers per dual lane.

Based on the information shared by Schiphol, the average wage of a security officer together with the extra

costs the employer makes per employee, the costs of a full-time equivalent (FTE[1]) security officer is equal €88,332. The number is higher due to the fact that Schiphol is not the direct employer of the security officer but hires them through security companies. The calculation on which this number is based is found in Table 6.1. The monthly salary in this table is based on an hourly (all-in) wage of €42.50.

Table 6.1: Calculation to estimate the costs of one security officer FTE.

The hourly cost of a security officer (Incl. all expenses)	€42.50
Hours per week	40
	×
Monthly cost of a security officer	€7,361
Months per year	12
	×
Estimated cost of a security officer (One FTE)	€88,332

Unfortunately, due to security constraints, we cannot mention the performance of security officers or compare it with the network's performance. However, according to Section 4.1 the algorithm cleared the CONF% detection baseline with a false alarm rate below 1%. Before the algorithm is able to replace the officers completely, there will be a transition period where the algorithm will be able to assist the security officer in their decision making. A model with the performance mentioned before is acceptable to properly assist an operator while not producing too many false alarms. Our baseline for all calculations in this section will be based on a dual lane with 11 security officers. This first phase (operator assist) will lead to faster decision times and for the third monitoring security officer to become superfluous. This results in a €88,332 costs reduction, which is calculated in Table 6.2.

Table 6.2: Cost savings per FTE with the algorithm as support and full replacement.

dual lane costs before implementing any algorithm	€88,332 * 11 FTE =€971,652
dual lane costs after implementing the algorithm as support	€88,332 * 10 FTE =€883,320
	-
1 FTE Costs saved:	€88,332
dual lane costs before implementing any algorithm	€88,332 * 11 FTE =€971,652
dual lane costs after implementing algorithm stand-alone	€88,332 * 8 FTE =€706,656
	-
3 FTE Costs saved:	€264,996

After a successful testing period and the potential product being fully developed, the option to replace security officers will occur[2]. With our baseline of 11 security officers, a fully automated system will reduce the number of security officers to 8. This will lead to a €264,996 or 27.3% costs reduction per dual lane per year. This calculation is displayed in Table 6.2.

The security officers that are not necessary anymore to do this repetitive job can be redeployed elsewhere. Due to this, it is possible for Schiphol to grow without the need to hire more security officers.

[1]Based on a 40-hour working week

[2]Before a fully automated system can be used, the detection and false alarm rates for a minimal viable product should be defined. After these definitions and guidelines, there are some changes to the law required to allow the use of fully automated systems at airports.

6.2.2 Implementation and maintenance cost estimation

This estimation is based on internal numbers of similar purchases. The final product will most likely be delivered on a separate machine since the classification or detection algorithm requires a dedicated GPU for classifying 3D images. This machine will cost around €20,000 to €40,000 whether or not the airport will showcase[3] the product. Schiphol will likely be (one of) the first airports to introduce these systems, aiming to showcase the product and will buy a large amount, resulting in a price per piece probably around €25,000. This price will likely include the algorithm, the machine which runs the algorithm and the installation to the point where it is fully functional. With 80 lanes requiring this upgrade, the total implementation cost will be around €2 million.

However, from the moment these systems will be in place, there are maintenance costs that need to be taken into account. These maintenance costs will be (with the assumption that the detection systems will only be bought from a single manufacturer), based on a centralized maintenance team. This team will take care of these tasks we identified (and perhaps more):

- Resolve issues of non-optimal performing systems (estimated 3 FTE)
- Monitor performance of the systems (in collaboration with independent parties, estimated 5 FTE)
- System security updates (estimated 1 FTE)
- Algorithm updates (detection of new threats, better performance, estimated 1 FTE)
- System upgrades (if necessary for updates, estimated 1 FTE)

At least 8 FTE will be needed to fill the irregular working scheme with some extra workforce when the airport is busy. These 8 FTE will be monitoring and resolving issues. Another 3 FTE will be needed for updates and upgrades. These employees need proper education, training, and very specific skills of which a yearly salary is estimated to be €160,000. The maintenance cost of all systems will be somewhere around €1,760,000.

6.2.3 Cost savings for all double lanes at security checkpoints per year

From internal statistics, we know that the total number of working hours per year in screening lanes is equal to 2,510,000. The total FTE calculated is 1,230 to keep the security checkpoints running in Schiphol and is based on a 40-hour workweek, these calculations are displayed in Table 6.3. Since we know the total number of hours worked we do not have to take the irregular[4] working hours of Schiphol into account.

[3] The first airport to fully implement these machines will serve as a showcase to other airports, who then might purchase
[4] Schiphol airport operates from 05:00 in the morning until 00:00 midnight, which results in 19 hours per day for 7 days per week.

Table 6.3: Calculation for the total amount of FTE needed to fill all security lanes for an average entire operational week.

Total security officer working hours per year (2018)	2,510,000
Yearly working hours per one FTE based on 40 hours per week	2,040
	÷
Total FTE	≈ 1,230

Next, the total labor costs on only the security lanes are calculated to be close to €110 million, which is also in Table 6.4. With the potential savings percentage calculated in Section 6.2.1 to be 27.3%, the potential cost reduction by implementing the algorithm with maintenance costs taken into account is **just over €28.5 million per year**. The complete calculation is found in Table 6.4, these are based on the reduced number of security officers necessary due to the implementation of a detection algorithm. For the first year, based on the estimation in Section 6.2.2, the implementation cost will reduce the potential savings to €26.4 million, the years after this number will be €28.4 million.

Table 6.4: Calculations for the potential savings for Schiphol after the first year.

Costs FTE security officer at Schiphol	€88,332
Total FTE	≈1,230
	×
Total costs security officer at Schiphol	≈€108,648,360
Potential savings with stand-alone algorithm	27,3%
	×
Potential total savings at Schiphol	≈€29,660,000
Extra maintenance costs (Section 6.2.2)	≈€1,760,000
	−
Potential total savings at Schiphol	≈€28,400,000

6.2.4 Passenger flow impact

In addition to the cost-saving impact, the introduction of an automated detection algorithm has also impact on the passenger flow. Right now, security officers take around 7-15 seconds to check the CT image of baggage, detect any potentially harmful items, and classify it as safe or unsafe. On the other hand, the algorithm is able to process 2880 images in just over 4 minutes, thus the classification of baggage will no longer be a bottleneck. By implementing a fully automated algorithm, the bottleneck of clearing a bag will no longer be present and the 10 second decision time build in the conveyor belt, will vanish. The next bottleneck of the process will be the speed of which the CT machine is able to capture a CT image.

However, such a possible system acceleration raises more future work research questions. Such as *"Is the CT machinery able to scan and produce images in such a short time?"* or *"Are the security officers able to keep up with resolving false alarms?"*. If the CT images check time is a bottleneck, by implementing an automated algorithm, this will eliminate the bottleneck and shift it to another part in the process.

6.3 Limitations

Limitations have emerged while conducting the experiments, writing this work, and talking with experts in the field. These limitations are divided into two sections: the limitations of the dataset in Section 6.3.1, and the limitations of the algorithm in Section 6.3.2.

6.3.1 Dataset

The dataset which is used to train, validate and test the model consists of CT images of real cabin baggage, which passed through a security checkpoint at some point in time. Since barely any of the bags in these images contain potentially harmful items, let alone contain firearms, the positively labeled data had to be artificially produced. The artificial insertion of isolated firearms leads to images without the artifacts that would be present in bags with real potentially harmful items. The model might be trained to detect without artifacts. Experiments are needed to make sure it will also work with real data. Otherwise, its performance on real-life baggage will drop significantly, compared to the reported performance on this dataset.

The firearms in the images of the positive dataset are not labeled with their location in the image. Therefore, object detection was excluded from this work and only classification experiments were conducted. The images were not manually labeled for this study, due to the need for baggage screening education and experience with detecting the potentially harmful items.

The images are only gathered from a single airport (Schiphol) and for a short period of time (one month), this results in a biased dataset. Namely, the passengers at Schiphol most likely pack their bags with different items than passengers at other airports. This can lead to unrepresentative false alarm rates due to the items packed by people who are less represented in the dataset appearing less often. This can also happen when the algorithm is trained on a dataset from one airport but is used at a different airport. Finally, the short gathering period can result in an abundance of business or leisure travelers, which again results in bias.

The size and complexity of the individual images are very high, and the number of images is rather low. Especially if compared to a less complex classification task like the MNIST data [54], which contains 60,000 training and 10,000 test images. A higher number of images, even if only negatively labeled, might result in a better performing model. Without extra images, the issue of a low number of images could be alleviated by using transfer learning. This method used the pre-trained weights from another model which is trained on a different, larger dataset. However, at this moment there are no weights to be transferred since there are no comparable trained algorithms for 3D images. Transfer learning could have been used for the CNN/RNN network but to keep the comparison equal, we have decided not to implement this.

6.3.2 Algorithm

The dataset was split in 80%-10%-10% subsets, as mentioned in Section 3.8.3. This split is stratified, meaning there is the same positive/negative labeled image ratio in any subset.

Unfortunately, the type of firearm in a bag is not extracted from the data. For this experiment, we did not extract all images from a single type of firearm. If this would have happened, the extraction of a number of firearm models from the dataset would demonstrate whether the network is able to generalize firearms or that it overfitted on certain firearm types

At this moment, particular slices that are fed into the CNN/RNN network are sliced in the x dimension. The slice is a 2D representation in the y-, and z-axis of our 3D model, there is no reason why that is. There is a chance the model works better in other or multiple directions. With zero padding preprocessing (Section 3.5.4) in all directions, resulting in a cubic image, the images can be classified using slices in multiple directions. This could lead to better results, but again, these experiments should be conducted in future work.

The model is trained to classify images which contain a firearm and does not detect[5] firearms. The reason for this is the dataset limitations discussed in Section 6.3.1. This way we are unable to determine whether the algorithm classifies an image as potentially harmful because of the firearm or another item. This is because the algorithm might be trained on items which are uniquely found in bags with firearms. When the model is trained this way, finding these items in harmless bags, the false alarm rate will be higher. There will also be a lower recall for harmful bags without these items.

The limitations of the algorithm arise from the dataset issues. Since the dataset does not contain localization information, the algorithm is unable to produce results for certain types of firearms. Also, insights in what firearms are hard to detect, and which are easier, are not produced. This could help with a clearer understanding of which items more sample images are needed. Thus, for the next data gathering, these points should be taken into account and can remedy some limitations. In summary, a solution for the mentioned limitations is a complete and fully labeled dataset, which is equally sampled from subgroups of passengers. These subgroups are variables like destination, origin, business/leisure, etc.

6.4 Future Work

This sections will discuss the future work discovered while writing about and conducting the experiments in this work. It is partitioned into three subsections. Section 6.4.1 will discuss possible research towards the improvement of the algorithm or the dataset that was used to run the experiments. Section 6.4.2 discusses the

[5]First detecting objects and then classifying those objects

possible adversarial learning experiments to be conducted. Section 6.4.3 will discuss the other, more general discovered future subjects.

6.4.1 Algorithm and data

While the experiments that were conducted in this work were thoroughly tested, it only applies to this network and only addresses binary classification (firearm/no-firearm). Although it is useful to have an algorithm to detect firearms in passenger cabin baggage, there are other items prohibited on planes [69]. This section will go over possible upgrades of the algorithm proposed in this work, mainly intending to provide the security officer who has to search through the bag with valuable information about the items contained in the bag.

Research could be conducted towards the classification of bags with other prohibited items like knives or blunt harmful items. If these also produce promising results, an algorithm could be developed to classify bags as safe or unsafe, based on any of the items on the prohibited items list of Schiphol [69]. Unsafe bags refer in this case to any bag containing some sort of prohibited item.

As a security officer at Schiphol Airport, not knowing exactly what type of potential harmful item is present in a bag, may result in longer searching times, due to searching through the CT image manually for all types of prohibited items. Therefore the use of a multi-class classification algorithm should be researched. This algorithm would use multiple classes while being trained to classify a bag as “firearm”, “blunt”, “knife”, or “clean” according to its contents.

The next step is, besides the classification of an object, to determine the location of the object, which is an object detection approach. For example, the use of the faster R-CNN algorithm [70] could be adapted to a 3D algorithm and color the detected objects. Within the detection task, there should also be research conducted towards the possibilities of bags containing multiple potential harmful items. In this case, an algorithm can be developed to detect and classify multiple objects and report “2 knives” or “1 blunt and a firearm”. This upgrade will result in more complete information for the security officer and hopefully help towards faster search times.

The multi-class classifier, the multi-object binary classifier, and the combined binary classifiers should be compared to each other based on both operator assist and automated screening. At this moment, the preferred requirements need to be decided. The following questions should be asked to both the government, who writes the policies, as well as the company, that carries out the algorithm:

- What is the minimum acceptable cut-off point between detection rate (Recall) and false alarm rate?

- How many potentially harmful items should be found for every bag opened? or: How many bags should be opened at most to find one potential harmful item? (Precision)

- What is preferred; an algorithm with relatively high performance but providing less information, or an algorithm with relatively low performance but providing more information? (Combined performance)

Each of the next steps mentioned above requires more precisely labeled data. The gathering of the positive samples and the corresponding labeling should start as soon as possible, to stay ahead of the demand and not bottleneck the production of newer, more precise algorithms. Even though transfer learning was not used in this experiment, it can be used with the CNN/RNN network. This can reduce the necessary number of images in the dataset greatly, however this must be researched.

Quality monitoring

When the limitations of this work are resolved and a neural network model is properly trained and being used to the classification task at Schiphol, the monitoring task starts. Monitoring is necessary to ensure the quality of the system. Because of people with bad intent trying to conceal items, uniquely appearing potentially harmful items will most likely be found while the algorithm is in use. But even a new type of suitcase might confuse the algorithm, which needs to be acted on.

To ensure proper screening, the flaws of the algorithm, as well as the speed, should continuously be monitored. The flaws of the algorithm can be monitored while in the operator assist phase. For example, when human security officers classify a bag as containing a potentially harmful item while the algorithm does not. The bag is opened and the actual outcome is used as feedback for the algorithm. In the automated phase, continuous testing of the algorithm by sending harmful bags through the machines would be a possible quality monitoring method. Based on a testing period before using the algorithm, the stability of the system can be tested. These tests should be designed and executed by an independent party.

Besides monitoring of the current model, the dataset should be extendable with newly scanned baggage images, to be trained and tested and finally be able to be detected. Therefore, research is necessary to discover optimal update-methods for neural networks in a highly volatile work environment as Schiphol. For example, incremental learning can be implemented, where a new model is trained based on a new batch of data and the previous model. The main concern is the need to prevent the model from forgetting what it has learned before (catastrophic learning).

6.4.2 Adversarial Learning

The adversarial learning section in this work is based on related work projected on our research. Unfortunately, there are no real comparable experiments to our 3D data. This results in derived assumptions based on the related work. These assumptions should be tested by designing adversarial networks that attack a network like the one proposed in this work. The "patch"-like, and 3D printed object should be generated and produced according to known methods from attacks on 2D classifiers. If these objects show significant opportunities, research towards matching countermeasures should be conducted.

6.4.3 General neural network usage

If adversarial learning turns out not to be an issue, and neural networks are used as an operator assist or an automated detection system, continuous monitoring of the performance of the algorithm is necessary. This is to keep track of its wrongly classified images and update the algorithm whenever needed by retraining with the newly available data and tuning hyperparameters accordingly. This is often a time-consuming job for which employees need a high knowledge base of deep learning, the algorithm itself, and the field in which it is used.

It is only a matter of time before adversarial learning attacks will be easily available and usable for CT machines. Even if currently not a threat new types of attacks may be introduced in the future. These developments should be closely monitored to prevent leaks.

Chapter 7

Summary & Conclusion

This work documents the research conducted on the automated detection of potentially harmful items in cabin baggage task using supervised classification algorithms. The focus has been on the 3D CT image classification, where others in this field only conducted experiments using 2D imagery. We designed and ran experiments on two networks, which are both able to process the 3D data.

The first algorithm uses the AlexNet CNN structure but all layers use spatial convolutions over the 3D baggage image. It accepts a 3D input and a 3D kernel is moving in all three dimensions. The number of units in each layer is the same as the original AlexNet, and also the kernel and stride values are the same (besides having a third dimension). Only the final two fully connected layers have been adapted to 2048 and 512 units, due to the lower number of output units.

The second algorithm is based on a feature-extracting-CNN in combination with a RNN combining the features of the 2D slices. This design accepts series of 2D slices of which the features are first extracted by the convolutional layers of the original AlexNet. Then, the extracted features are fed into two LSTM layers combining the information of multiple slices, resulting in a safe/unsafe classification. This network also uses the standard AlexNet filter, kernel, and stride values. The LSTM layers have 128, and 64 units feeding a 256 unit fully connected layer.

To keep the comparison equal, both algorithms use the same Adam optimizer with a 10^{-4} learning rate, and a binary cross-entropy cost function. Both algorithms are 10-fold cross validated, ran three times, and the folds are equal for both experiments.

Based on the results in Section 4 we can say that, with these hyperparameters, the CNN/RNN algorithm outperforms the AlexNet-like algorithm, while this might change if both algorithms' hyperparameters are optimized or more training data becomes available. With a recall value of 90% we find a precision value of

76.5%, and a FATP value of 2.5%. This translates in the following statements:

- *"For all images of baggage containing firearms, 90.0% were successfully classified as unsafe."*
- *"For all images of baggage that were classified as unsafe and thus should be opened, 76.5% actually contain a firearm."*
- *"For all baggage that is run through the system, 2.5% will be wrongly classified and opened without containing a firearm."*

When comparing the results in this work to that of Akcay *et al.* [1] it looks like their 2D approach outperforms our proposed classifier, however, due to the different dataset, we are not able to directly compare the two. Like Akcay's approach, it looks like our classifier also outperforms the visual bag of words method mentioned in their work.

Considering the upcoming technological progression in computational power, a possible higher quality dataset, and our results, we think there is potential in using 3D classifiers for automated detection. At this moment, an operator assist system is feasible, which might improve the speed of screening agent without significantly increasing the number of false alarms. This may lead to improved development of better performing algorithms.

With the upcoming progression we also took a look at the opposing parties who might use some sort of adversarial learning. There are quite a few adversarial algorithms developed in the past year. These algorithms are designed to mislead neural networks and might be used by malevolent people. Due to the nature of our data (3D in Hounsfield units), feature squeezing and defensive distillation will reduce the risk. However, the progression on the adversarial learning field should be kept in mind, to stay ahead of such attacks.

Bibliography

[1] S. Akcay, M. E. Kundegorski, C. G. Willcocks, and T. P. Breckon, "Using deep convolutional neural network architectures for object classification and detection within x-ray baggage security imagery," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2203–2215, 2018.

[2] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[3] V. Lugade, "Neural networks a multilayer perceptron in matlab," Dec 2012.

[4] C. Olah, "Understanding lstm networks," Aug 2015.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[7] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," *arXiv preprint arXiv:1712.09665*, 2017.

[8] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1625–1634, 2018.

[9] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," *arXiv preprint arXiv:1707.07397*, 2017.

[10] C. B. of Statistics (CBS), "Over 76 million passengers travel via dutch airports." https://www.cbs.nl/en-gb/news/2018/05/over-76-million-passengers-travel-via-dutch-airports, jan 2018. The amount of travelers using dutch airports.

[11] "Are you flying via schiphol in the may holiday period? this is what you need to know!." https://news.schiphol.com/are-you-flying-via-schiphol-in-the-may-holiday-period-this-is-what-you-need-to-know/, Apr 2018.

[12] R. Z. Herman Grech, "Plane hijack drama in malta ends; all hostages released," *Times of Malta*, dec 2016. News article about the most recent hijack due to a dangerous item passing security.

[13] A. S. N. (ASN), "Hijacking description: Afriqiyah airways flight 8u209." https://aviation-safety.net/database/record.php?id=20161223-0, dec 2016. Aviation Safety Network database about Afriqiyah Airways flight 8U209, the most recent hijack due to a dangerous item passing security.

[14] M. van Justitie en Veiligheid, "Nationaal coördinator terrorismebestrijding en veiligheid (nctv)."

[15] "New scanners to reduce need for unpacking at airport security." https://www.reuters.com/article/us-europe-airport-security/new-scanners-to-reduce-need-for-unpacking-at-airport-security-idUSKBN13I1BB, Nov 2016.

[16] M. Baştan, M. R. Yousefi, and T. M. Breuel, "Visual words on baggage x-ray images," in *Computer Analysis of Images and Patterns*, pp. 360–368, Springer Berlin Heidelberg, 2011.

[17] M. Baştan, "Multi-view object detection in dual-energy x-ray images," *Machine Vision and Applications*, vol. 26, pp. 1045–1060, aug 2015.

[18] S. Healthineers, "Radiography (plain x-rays)."

[19] D. Attwood and A. Sakdinawat, *X-rays and extreme ultraviolet radiation: principles and applications*. Cambridge university press, 2017.

[20] V. Rebuffel and J.-M. Dinten, "Dual-energy x-ray imaging: benefits and limits," *Insight-non-destructive testing and condition monitoring*, vol. 49, no. 10, pp. 589–594, 2007.

[21] J. Radon, "1.1 über die bestimmung von funktionen durch ihre integralwerte längs gewisser mannigfaltigkeiten," *Classic papers in modern diagnostic radiology*, vol. 5, p. 21, 2005.

[22] "Hounsfield scale," Jan 2019.

[23] S. . D. S. I. L-3 Communications, "Clearscan — cabin baggage screening — providing advanced cabin baggage screening solutions and carry-on baggage scanning solutions with ct technology."

[24] T. R. Johnson, "Dual-energy ct: general principles," *American Journal of Roentgenology*, vol. 199, no. 5_supplement, pp. S3–S8, 2012.

[25] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, pp. 1150–1157, Ieee, 1999.

[26] T. Franzel, U. Schmidt, and S. Roth, "Object detection in multi-view x-ray images," in *Lecture Notes in Computer Science*, pp. 144–154, Springer Berlin Heidelberg, 2012.

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

[28] D. Turcsany, A. Mouton, and T. P. Breckon, "Improving feature-based object recognition for x-ray baggage security screening using primed visualwords," in *2013 IEEE International Conference on Industrial Technology (ICIT)*, IEEE, feb 2013.

[29] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*, pp. 404–417, Springer, 2006.

[30] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 5, pp. 433–449, 1999.

[31] V. Riffo and D. Mery, "Automated detection of threat objects using adapted implicit shape model," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, pp. 472–482, apr 2016.

[32] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *International journal of computer vision*, vol. 77, no. 1-3, pp. 259–289, 2008.

[33] D. Mery, E. Svec, M. Arias, V. Riffo, J. M. Saavedra, and S. Banerjee, "Modern computer vision techniques for x-ray testing in baggage inspection," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 4, pp. 682–692, 2017.

[34] D. Mery, V. Riffo, U. Zscherpel, G. Mondragón, I. Lillo, I. Zuccar, H. Lobel, and M. Carrasco, "Gdxray: The database of x-ray images for nondestructive testing," *Journal of Nondestructive Evaluation*, vol. 34, no. 4, p. 42, 2015.

[35] N. Jaccard, T. W. Rogers, E. J. Morton, and L. D. Griffin, "Tackling the x-ray cargo inspection challenge using machine learning," in *Anomaly Detection and Imaging with X-Rays (ADIX)* (A. Ashok, M. A. Neifeld, and M. E. Gehm, eds.), SPIE, may 2016.

[36] A. Mouton and T. P. Breckon, "A review of automated image understanding within 3d baggage computed tomography security screening," *Journal of X-ray science and technology*, vol. 23, no. 5, pp. 531–555, 2015.

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[40] A. Wales, T. Halbherr, and A. Schwaninger, "Using speed measures to predict performance in x-ray luggage screening tasks," in *43rd Annual 2009 International Carnahan Conference on Security Technology*, IEEE, oct 2009.

[41] E. Hallstrm, "Introduction to tensorflow - cpu vs gpu," Nov 2016.

[42] F. Chollet *et al.*, "Keras." "`https://keras.io`", 2015.

[43] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[44] T. Oliphant, "NumPy: A guide to NumPy." USA: Trelgol Publishing, 2006–. [Online; accessed ¡today¿].

[45] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online; accessed ¡today¿].

[46] T. M. Mitchell *et al.*, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, vol. 45, no. 37, pp. 870–877, 1997.

[47] M. Minsky and S. Papert, "Perceptron expanded edition," 1969.

[48] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[49] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

[50] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2018.

[51] A. Cauchy, "Méthode générale pour la résolution des systemes déquations simultanées," *Comp. Rend. Sci. Paris*, vol. 25, no. 1847, pp. 536–538, 1847.

[52] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[54] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[55] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2017.

[56] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5353–5360, 2015.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[58] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently recurrent neural network (indrnn): Building a longer and deeper rnn," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5457–5466, 2018.

[59] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[60] "Dicom standard."

[61] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[62] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *ICLR 2017*, 2017.

[63] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[64] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," *arXiv preprint arXiv:1804.08598*, 2018.

[65] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, IEEE, 2016.

[66] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, "Learning with a strong adversary," *arXiv preprint arXiv:1511.03034*, 2015.

[67] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[68] I. Goodfellow, P. McDaniel, and N. Papernot, "Making machine learning robust against adversarial inputs," *Communications of the ACM*, vol. 61, no. 7, pp. 56–66, 2018.

[69] "List of items prohibited on a plane."

[70] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.