Master Thesis
Business Analytics

Comparative study on object detection models and augmentation techniques for face detection on camera-captured identity documents

Fay Westendorp (2617778)

April 26, 2023

Digital Power,
Your Data Partner

H.J.E. Wenckebachweg 123
1096AM Amsterdam

Company Supervisor:
Joachim van Biemen

Vrije Universiteit
Amsterdam,
Faculty of Science

De Boedelaan 1081a
1081HV Amsterdam

Supervisor:
Prof. Dr. Mark Hoogendoorn
Second reader:
Prof. Dr. Ger Koole

Vesteda,
Huurwoningen in
heel Nederland

De Boelelaan 759
1082RS Amsterdam

Company Supervisors:
Dior Adel
Ruben Sikkes

Preface

This thesis aims to fulfil the requirements of the Master of Business Analytics program at Vrije Universiteit Amsterdam. The Business Analytics program introduces a combination of computer science, mathematics, and business management techniques to aid in identifying and resolving business issues. This thesis aims to merge academic research with practical solutions to help the internship company and develop my expertise in this study's field. This research mainly focuses on the possibility of detecting faces on camera-captured identity documents.

I would like to express my gratitude towards Prof. Dr. Mark Hoogendoorn, my academic supervisor, for his guidance, support, and interest throughout this internship. I would also like to thank Prof. Dr. Ger Koole for serving as the second reader.

My internship took place at the housing corporation Vesteda via Digital Power, a data consultancy company. From Digital Power, I would like to thank Joachim van Biemen, my external supervisor, for his enthusiasm and assistance during this internship. Similarly, I thank all my colleagues at Digital Power for their advice, support, and the great atmosphere at the office. It all made my time at Digital Power very enjoyable! From Vesteda, I would like to thank Dior Adel and Ruben Sikkes for their continuous interest and assistance throughout this internship.

Lastly, I would like to thank my parents for providing me with their unfailing support throughout my years of study, specifically through the process of writing this thesis.

Abstract

Due to the significantly growing amount of personal data stored online by users, businesses must handle this incoming flow of sensitive information as adequately as possible. However, companies often struggle to comply with changing data security regulations due to challenges or unawareness of requirements. To comply with strict data privacy and security regulations, companies request their users to black out any sensitive information that is unnecessary for document usage. Although this method can be successful, its effectiveness heavily depends on users following instructions, and not all may comply with the provided guidelines. Alternatively, companies may opt for manually detecting sensitive information by employees. While this method may prove effective, the process is time-consuming and disadvantageous to the company.

To address these challenges, this research proposes a comparative analysis of two object detection models for face detection on camera-captured identity documents, focusing on the face as a sensitive information part present on any identity document. The two object detection models employed in this research are Faster R-CNN and RetinaNet, both with a ResNet-101 backbone and FPN. Furthermore, to augment the limited available data to train on and assess the effectiveness of data augmentation techniques, three transformations and a combination of all are applied to the training data: a horizontal flip, a random crop, and colour jittering. Additionally, both models are evaluated without any augmentation to enable a conclusive evaluation of the performance of all techniques. The comparative study on object detection models and data augmentation techniques will provide more knowledge on face detection on camera-captured identity documents, which will be useful for businesses across all fields in processing large amounts of personal data quickly and efficiently while adhering to data privacy and security regulations.

According to the results of the comparative study, RetinaNet produced more accurate predictions than Faster R-CNN, as determined by various average precision metrics and false positive and false negative rates. Additionally, the RetinaNet model, when combined with random crop augmentation, yielded the best results based on the evaluation of false positive and false negative rates. In conclusion, this research recommends that businesses such as Vesteda use the RetinaNet model with random crop augmentation to detect and ultimately blur faces effectively on camera-captured identity documents submitted by potential tenants.

Contents

List of Abbreviations

ANN *Artificial Neural Network*. 5, 7

AP *Average Precision*. 33–36, 46

CNN *Convolutional Neural Network*. 3, 8, 11–15, 17

Conv *Convolutional Layer*. 8, 9, 15, 23, 24, 27, 28

DL *Deep Learning*. 5

DSSD *Deconvolutional Singe Shot Detector*. 15, 16

FC *Fully Connected Layer*. 10

FCN *Fully Convolutional Network*. 14, 15

FPN *Feature Pyramid Network*. 14–17, 22–25, 27

GDPR *General Data Protection Regulation*. 1, 3

HOG *Histogram of Oriented Gradients*. 12

ILSVRC *ImageNet Large Scale Visual Recognition Challenge*. 10, 11

IoU *Intersection over Union*. 10, 15, 26, 32, 34, 35, 37, 44, 45

mAP *Mean Average Precision*. 2, 10, 13–17, 33

ML *Machine Learning*. 3, 5

MSE *Mean Squared Error*. 7

NMS *Non-Maximum Suppression*. 26

R-CNN *Region Proposal Convolutional Neural Network*. 13–16, 22, 25–28, 32, 33, 35–42, 44, 46

ResNet *Deep Residual Network*. 23–25, 27

RoI *Region of Interest*. 26, 27, 32

RPN *Region Proposal Network*. 13, 14, 22, 25–27, 32, 38

SIFT *Scale-Invariant Feature Transform*. 12

SSD *Single Shot Detector*. 13, 15, 16, 35

SVM *Support Vector Machine*. 4, 14

YOLO *You Only Look Once*. 13, 15, 16, 19, 35, 44

Chapter 1

Introduction

In the current era of digitization that characterizes our society to date, the amount of personal data stored online is growing significantly. The increasing number of online processes extend the spread of personal information across all types of businesses, such as banks, online retail, and residential companies. These businesses must handle the constant flow of personal information as adequately as possible, ensuring sensitive data is securely maintained in protected environments to minimize the risk of leaks and fraud. However, companies often struggle to comply with changing data security regulations such as the General Data Protection Regulation (GDPR) [1]. They may not be adequately prepared for the challenges and could be unaware of the requirements stated by such legislation, possibly resulting in major complications such as privacy violations, unlawful storing of data, or identity fraud. The identity document is an example of personal information often uploaded by users to participate in processes online. This document should be handled with great care by a company gathering personal information from users since it contains large amounts of sensitive information.

In order to adhere to the strict regulations on data privacy and security, companies ask their users to black out any sensitive information present on such documents that are not required to be visible. While this approach can be effective, it relies heavily on users to follow instructions, and not everyone may adhere to the guidelines. Alternatively, companies may introduce the manual detection of sensitive information by employees, followed by a request for the user to re-upload a correct document version. While this approach may work accordingly, the process is considered very time-consuming and disadvantageous for the company. To address these challenges, this research proposes a comparative analysis of object detection models for face detection on camera-captured identity documents, focusing on the face as a sensitive information part present on a personal identity document. Such detection models will detect the location of each face present on an identity document making it possible to black out this part of the document or make it illegible. Existing literature on face detection typically encompasses datasets that contain images captured across diverse environmental conditions [2], while previous studies on the detection of identity documents mainly include classification [3], and the detection of identity documents as a whole [4]. Therefore, this research combines both by introducing face detection specifically for camera-captured identity documents to address that challenge accordingly. Despite deep learning models such as object detection models significantly outperforming traditional machine learning methods on the task of face detection, they show poor performance when evaluated on datasets containing a limited number of instances [5]. Therefore, in addition to the comparative study on object detection models, various data augmentation techniques will be utilized to augment the available data as well as to assess the effectiveness of these different augmentation techniques.

1.1 Research Questions

To effectively address the challenges above, this research is centred around the following main research question:

RQ: *How accurately can object detection models detect faces on camera-captured identity documents?*

The following sub-questions are formulated to give a conclusion with regard to the main research question. Each sub-questions addresses one of the challenges mentioned above.

SQ 1: *Which object detection models in the literature effectively detect faces in identity documents?*

Rather than attempting to develop a novel technique, several detection models from previous work in related fields are chosen for comparison in this research. Model performances are compared through various metrics, including the mean average precision (mAP) and false positive and false negative rates to answer this sub-question.

SQ 2: *How effectively will various data augmentation techniques influence the robustness of a detection model? Will a combination of multiple yield a different effect?*

This research features a limited dataset and includes augmentation techniques to broaden the available data further. Additionally, several augmentation techniques will be compared to determine their influence on the performance of the object detection models. To answer this sub-question, the detection models are combined with several different data augmentation techniques as well as the combination of all to analyze and compare their performance. Furthermore, the models are also fine-tuned without any augmentation to enable a conclusive evaluation of the performance of all augmentation techniques. Finally, the performances are compared through the metrics mentioned above.

1.2 Report Structure

In Chapter 2, several subjects relevant to the topic of this thesis are broadly explained. In Chapter 3 relevant literature is reviewed. Chapter 4 provides an analysis of the dataset used for the comparative study. Thereafter, Chapter 5 gives a description of the implemented models and augmentation techniques and is followed by Chapter 6, the experimental setup. Chapter 7 presents the analysis of the results. Finally, chapter Chapter 8 discusses the limitations and possibilities for future research and Chapter 9 concludes this research by answering the main research question and its sub-questions and providing a recommendation.

Chapter 2

Background

This chapter clarifies necessary knowledge about several subjects closely related to the research question. First, the internship company Vesteda is introduced. Thereafter, the different types of machine learning are specified and explained, followed by a description of deep learning. Furthermore, the structure and utilization of an artificial neural network often used in deep learning, as well as its fundamental building block, the backpropagation algorithm, are formulated. The components of a Convolutional Neural Network (CNN), a specific type of neural network generally used for image datasets, are summarized. Last, several milestones in the ImageNet Competition are discussed due to their essential role in how object detection models evolved.

2.1 About Vesteda

Vesteda is a Dutch residential investor focusing primarily on the mid-rental segment in urban and economically strong regions. They invest funds for institutional investors such as pension funds and insurers and hold a portfolio of over 27,500 homes. Every month, approximately 300 apartments become available for rent, for each of which an estimated 20 applicants express interest in the apartment. Every new possible tenant registering at Vesteda must upload several personal documents before they can sign up for an apartment visit. One requirement is a copy of the identity document, such as an ID or passport. With this in mind, one can conclude that Vesteda owns large amounts of personal information from all their current, possible future, and past tenants. Handling this substantial amount of data poses some challenges for Vesteda. All collected identity documents should be handled carefully since they contain large amounts of sensitive information. Moreover, with the increasing emphasis on data privacy and security, it is important to strictly adhere to regulations such as the GDPR legislation. To comply with these regulations and minimize the amount of sensitive information present on every identity document while still making sure all necessary parts are visible, Vesteda asks their potential tenants to black out various parts of their identity document before sending it in.

2.2 Machine Learning

Machine Learning (ML) is a subfield of artificial intelligence that enables computers to learn and make predictions and decisions based on data using algorithms. A model will be trained on a dataset, allowing the model to then make predictions and choices based on what has been learned during training. Machine learning can be divided into various learning types, with the first being supervised learning. In this form of learning, a model is trained on a dataset that contains labelled instances, where the corresponding target values are provided for all instances in the training and test set. After training, the model can predict the label for unseen samples [6]. Examples of supervised learning are predicting customer churn, classifying sentiment in reviews, or identifying objects in images. In the case of unsupervised learning, a model is trained on an unlabeled dataset, where the correct target values for all instances are absent, and the model must find patterns and relationships within the data [7]. Examples of unsupervised learning are customer clustering based on characteristics and identifying patterns in stock price fluctuations. Semi-supervised learning is a variation of supervised and unsupervised learning where a model is trained on a partially labelled dataset, allowing the model to learn from both labelled and unlabeled instances [7]. Examples of semi-supervised learning are document classification on labelled and unlabeled documents or predicting disease types based on labelled data and unlabeled medical records. Last, in reinforcement learning, an agent is trained to maximize reward by taking a series of actions in an environment and learning through trial and error, translated into rewards and negative rewards. Examples of reinforcement learning are robotics training, optimizing manufacturing processes, or teaching a machine how to play a game of chess [8].

The extensive domain of machine learning results in a broad application of its capabilities throughout various fields, including object detection. Most object detection studies are presented as a supervised learning problem where bounding boxes around all objects present in each image serve as the labels of the dataset. Focusing on the different supervised machine learning algorithms types, Osisanwo et al. [9] elaborates on a broad selection. First, linear classifiers classify input based on the value of a linear combination of features and are often used when classification speed is considered important since linear classifiers are rated the fastest. Different types of linear regression include simple and multiple linear regression. Logistic regression is a classification algorithm that predicts a binary or multi-class output by modelling the probability of a particular outcome based on the input variables. It is commonly used in predicting the likelihood of an event happening. Another supervised learning algorithm is Naive Bayes which assumes features are independent of each other and calculates the probability of a certain outcome based on the input variables. It is a simple and efficient classifier that can handle high dimensional data and is commonly used in text classification and sentiment analysis [9]. Next, Support Vector Machines (SVMs) can also handle high dimensional data and are therefore used in both linear and non-linear problems. SVMs try to find the hyperplane that best separates the data into different classes by finding the maximum margin between the hyperplane and the closest data points of each class [9]. Furthermore, Decision Trees recursively split a dataset into smaller subsets based on the most significant feature and can be used for classification and regression problems. Random Forest is an ensemble learning method that consists of many decision trees. Last, Neural Networks can perform several regression and classification tasks simultaneously and depend upon three fundamental aspects: input and activation functions, network architecture, and the weights of each input connection. They require rather large amounts of data and computation power to train and are commonly used in image recognition and natural language processing.

In traditional machine learning approaches, before the rise of deep learning, feature engineering was often a crucial step in the modelling pipeline, as it can significantly affect the model's performance. Feature engineering is the process of selecting, extracting, and transforming raw input data into a more suitable format for training a machine learning model. The goal of feature engineering is to create a set of input features that can effectively represent the underlying patterns and relationships in the data, which can be learned by the neural network model to make accurate predictions [10]. However, with the rise of deep learning and the availability of large amounts of data, it has become increasingly common to use neural networks to learn features directly from raw input data without manual feature engineering. For example, the adaption of machine learning for object detection generally includes deep learning models, which are considered a type of neural network with many hidden layers. This deep network structure enables it to learn more complex patterns in data, making it suitable for object detection tasks.

2.3 Deep Learning

Deep Learning (DL) is a subfield of ML that utilizes artificial neural networks with a large number of hidden layers to learn and make intelligent decisions. An artificial neural network comprises multiple layers of interconnected nodes, including an input layer receiving raw data, an output layer producing a model's final output, and one or more hidden layers used to extract features and recognize patterns within the data. The ability to learn complex patterns and extract features from data is one of the key advantages of deep learning models. As a result, it establishes its success in a wide range of applications, including natural language processing, speech recognition, and object detection.

2.3.1 Artificial Neural Networks

An Artificial Neural Network (ANN) can be described as an interconnected assembly of simple processing units, i.e., neurons. The processing ability of the network is stored in the interunit connection strengths, i.e., weights, and obtained by the process of adaption to, or learning from, a set of training patterns [11]. Figure 2.1 shows a schematic structure of an ANN consisting of an input layer, two hidden layers, and an output layer. Starting from the input layer, each neuron receives input from several other neurons in the previous layer and uses this input to compute and output a signal to the connected neurons in the successive layer. For example, in Figure 2.1, the weight w_{mr} represents the strength of the connection between the value of neuron x_m in the input layer and the value of neuron h_r in the next layer of the network. By adjusting the weights and biases of the connections between neurons and the parameters of the activation functions, a neural network can learn to perform a wide range of tasks, particularly tasks that involve recognizing complex patterns and relationships.

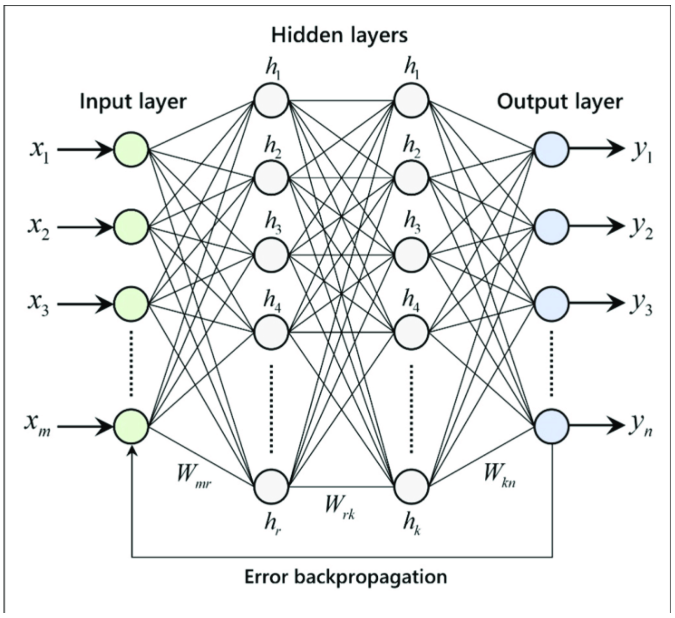


Figure 2.1: *Artificial Neural Network with Three Hidden Layers [12]*

For further clarification, Figure 2.2 shows a schematic structure of the operations done by a single neuron. In general notation, the weighted sum of the input of a specific neuron, denoted by $\sum W_i x_i$, is combined with a bias b and inserted into an activation function f, which gives the rescaled output of that neuron, denoted by y. Finally, M indicates the total number of input values connected to this specific neuron (Formula 2.1).

$$a = \sum_{i=1}^{M} w_i x_i + b \qquad \Longrightarrow \qquad y = f(a) \tag{2.1}$$

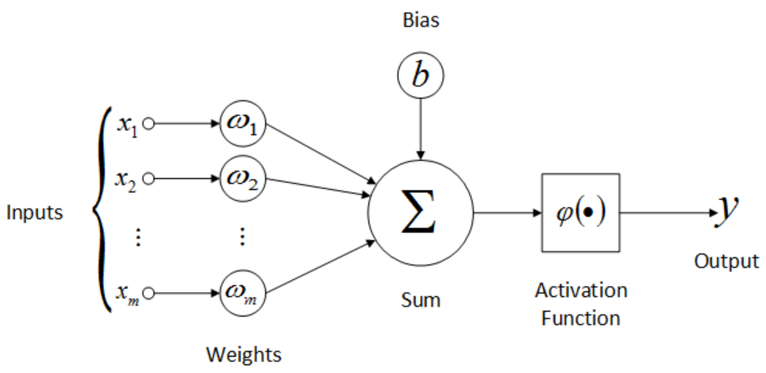


Figure 2.2: *Operations within a Neuron [13]*

There are several activation functions generally used in neural network architectures. The most common is the rectified linear unit (ReLU), which outputs the exact input when positive and zero when negative. Other commonly used activation functions include the sigmoid and hyperbolic tangent (tanh) functions. In every activation function shown below, the value of x represents the weighted sum of the input values, including the added bias.

$$ReLU(x) = max(0, x) \qquad Sigmoid(x) = \frac{1}{1+e^{-x}} \qquad Tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

2.3.2 Backpropagation

During the forward pass of a neural network, the input data is passed through the network layers to produce a prediction. The error between the predicted and actual output is then calculated using a loss function. An ANN uses backpropagation to minimize its error by traversing back through the network and letting the model learn from its mistakes. It works by propagating this error back through the network layers to update the weights based on how much they contributed to the error. This process is repeated iteratively until the network converges to a satisfactory level of performance. Backpropagation is a method to calculate the gradient of the loss function with respect to the weights and biases of the network. Subsequently, this gradient can be used to update the weights and biases using gradient descent. The process of backpropagation can be decomposed into four steps [14] [15] [16].

Feed forward propagation The input data is passed through the network, first encountering the hidden layers and last the output layer. As previously mentioned, given input vector x, the network calculates the output vector y by using the current weights and biases:

$$a^{(l+1)} = W^{(l)}x^{(l)} + b^{(l)}, for \quad l = 1, ..., L-1$$
$$z^{(l+1)} = f(a^{(l+1)}), for \quad l = 1, ..., L-1$$
$$y = f(a^{(L)},$$

where $a^{(l)}$ and $z^{(l)}$ are the input and output of layer l, respectively, $W^{(l)}$ is the weight matrix for layer l, $b^{(l)}$ the bias vector for layer l, f is the activation function and L the total number of layers in the network.

Loss calculation The loss L is determined by calculating the difference between the predicted output $\hat{y}$ and the true output y. Various types of loss functions can be utilized in machine learning, with one of the commonly employed ones being the Mean Squared Error (MSE). The MSE loss function is mathematically defined as:

$$L = \frac{1}{2}(y - \hat{y})^2$$

Backpropagation The next step is to perform backpropagation by calculating the gradient of the loss function with respect to the weights and biases of the network. Intuitively, backpropagation works by adjusting the weights and biases in the neural network to minimize the difference between the predicted and actual outputs. It does this by calculating the gradient of the error with respect to each neuron's weights and biases and then adjusting them in a direction that reduces the error. First, the error at each layer is calculated and propagated backwards through the network. The error ϵ is first calculated for the output layer (upper formula) and then for every hidden layer (lower formula):

$$\epsilon^{(L)} = (y - \hat{y}) \cdot f'(a^{(L)}),$$

$$\epsilon^{(l)} = (W^{(l)})^T \epsilon^{(l+1)} \cdot f'(a^{(l)}), for \quad l = L-1, ..., 1$$

where f' is the derivative of the activation function and $W^{(l)^T}$ is the transpose of the weight matrix of layer l. Last, the gradient of the loss functions with respect to the weights and biases, namely $\frac{\partial L}{\partial W^{(l)}}$ and $\frac{\partial L}{\partial b^{(l)}}$, are calculated:

$$\frac{\partial L}{\partial W^{(l)}} = \epsilon^{(l+1)} (z^{(l)})^T,$$

$$\frac{\partial L}{\partial b^{(l)}} = \epsilon^{(l+1)},$$

where $(z^{(l)})^T$ is the transpose of the output of layer l.

Weight and bias update The algorithm continues to iterate through the network, adjusting the weights and biases, until it finds the set of weights that produces the most accurate predictions.

2.4 Convolutional Neural Networks

A specific type of Neural Network often used when handling image data is a Convolutional Neural Network (CNN). This type is specialized in pattern recognition with a grid-like architecture, such as images. It is designed as a sequence of a combination of three possible layers: the convolutional layer, the pooling layer, and the fully connected layer, where all layers generally occur multiple times. Each neuron in the network processes the data in its receptive field, making sure simpler patterns such as lines, curves, and edges are detected first and more complex patterns such as faces and objects later subsequently [17]. A schematic picture of a CNN is shown in Figure 2.3.

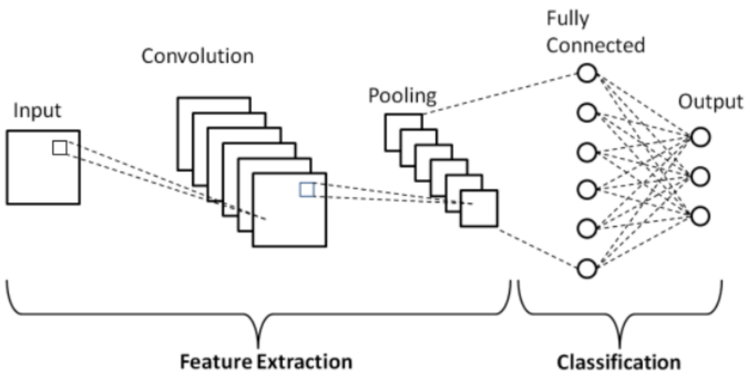


Figure 2.3: *Schematic picture of Convolutional Neural Network [18]*

Convolution Layer The convolution layer (Conv layer) carries the most significant part of the network's computational load. Therefore, it is considered the core building block of any CNN [19]. Goodfellow et al. discuss three important aspects of the convolution layer that motivate its use in applying the CNN [15]. First, the Conv layer has sparse interaction created by a filter size smaller than the input image. It uses the same set of filter weights across the entire input image, which reduces both the number of parameters and improves efficiency. A filter is a 2D weight matrix that captures a pattern or feature in the input data, such as an edge or a corner. Furthermore, the sliding filter of the Conv layer enables the detection of a feature regardless of the location of that feature on the image. Figure 2.4 shows a schematic representation of calculations done in the convolution layer.

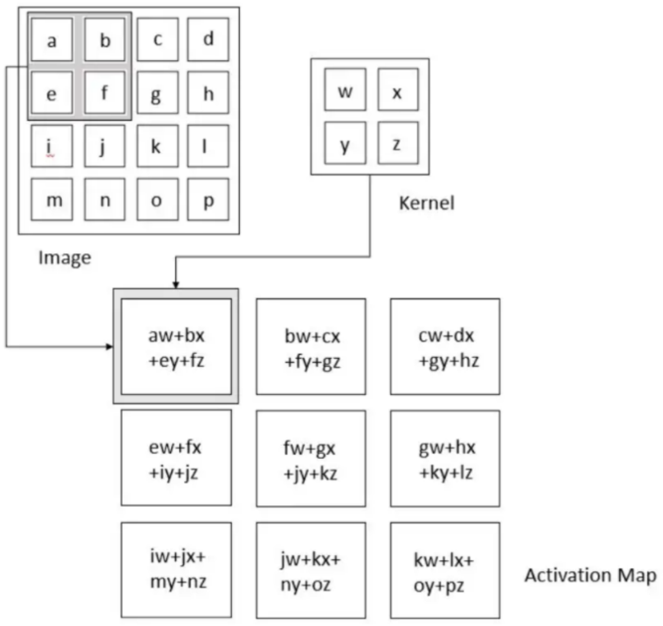


Figure 2.4: *Schematic representation of Convolution Layer [20]*

A filter or kernel (a set of learnable parameters) traverses over the image and performs the dot product between the kernel and a fraction of the total image resulting in a two-dimensional representation of the image, also known as the activation map or feature map. Let X be an image with dimensions H, W, D representing height, width, and depth, respectively. Let K be a kernel with dimensions kH, kW, D, kF, which represent the kernel's height, width, depth, and number of filters of the Conv layer, respectively. The output at position (i, j) of the activation map from the k^{th} filter can be written as [15]:

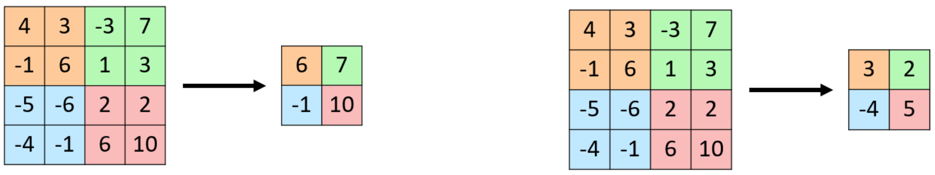
$$output_{i,j,k} = \sum_{a=0}^{kH-1} \sum_{b=0}^{kW-1} \sum_{c=0}^{D-1} X_{i+a,j+b,c} \cdot K_{a,b,c,f},$$

where *output* depicts the total activation map with dimensions mH, mW, mF representing the maps height, width, and number of filters, respectively. The height and width dimensions of the activation map can be calculated with the following formulas:

$$mH = \frac{H - kH + 2P}{sH} + 1, \qquad mW = \frac{W - kW + 2P}{sW} + 1,$$

where P is the amount of padding and S with dimensions sH, sW is the number of pixels the filter is shifted each time, also known as the stride.

Pooling Layer The pooling layer is able to derive a summary statistic of a fraction of input which reduces the number of training parameters and decreases the amount of computation and weights. This operation reduces the spatial dimensions of the input image while retaining important features. Different pooling functions are min pooling, max pooling, and average pooling, where the kernel's minimum, maximum and average value in each slice is passed onto the next layer, respectively. For max pooling, a pooling window traverses over the image with a certain stride and extracts the maximum value of each window, resulting in a new image with reduced dimensions. Figures 2.5a and 2.5b illustrate an example of max pooling and average pooling, respectively.



(a) *Max pooling* **(b)** *Average pooling*

Figure 2.5: *Example of max pooling and average pooling*

The calculation done by a max pooling window P with dimensions pH, pW can be formulated as:

$$output_{i,j,c} = \max_{a=0}^{pH-1} \max_{b=0}^{pW-1} X_{i \cdot S+a, j \cdot S+b, c},$$

where X is the input image, S is the stride of the window, and (i, j) the coordinates of the top left corner of the window. The output dimensions can be calculated with the formulas used in the convolution layer.

Fully Connected Layer All neurons in the fully connected layer (FC) have full connectivity with all neurons in the preceding and succeeding layer, similar to the neuron operation shown in Figure 2.2. The output value is calculated by matrix multiplication followed by an optional bias function and activation function.

2.5 Evaluation

In object detection, the most common evaluation metric is mean average precision (mAP), a metric that calculates the area under the precision-recall curve to measure the performance of an object detection model [21]. Precision is the fraction of true positive predictions among all predictions made by the model, whereas recall is the fraction of true positive predictions among all actual objects in the image. A detection is positive if the intersection over union (IoU) between the prediction and ground truth bounding box is higher than a certain threshold. The mAP evaluation metric will be further explained as a performance indicator in Chapter 6.

2.6 ImageNet Competition

Within the field of object detection, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [22] played a significant role in the establishment of state-of-the-art object detection models. The challenge aimed to evaluate and advance state-of-the-art object detection and image classification by providing a large-scale dataset, ImageNet, with millions of labelled images and a standardized evaluation protocol. From 2010 until 2017, the ImageNet challenge was hosted each year, enabling researchers to develop better models for computer vision tasks such as image classification, object localization, and object detection. The ImageNet dataset consists of over 1.2 million images and 1000 classes, partly annotated with labels and partly annotated with bounding boxes. Competitors were asked to correctly classify or detect different objects and scenes across subsets of the ImageNet dataset. Despite it being the world's largest labelled dataset of images publicly available on release, there was little interest in the dataset. Researchers were particularly focused on building better models by increasing performance and overlooked the benefit of having significant training data. This view permanently changed with the emergence of ILSVRC, and it became the primary benchmark for computer vision tasks.

During the first years of the ImageNet competition, simpler tasks such as image classification were part of the main challenges. In the 2013 challenge, a new object detection task was introduced, and later on, object localization and object detection in videos [22]. The ImageNet competitions initiated the use of deep neural networks, specifically CNNs, for image classification tasks, resulting in researchers developing these techniques, further leading to significant model improvements compared to traditional machine learning algorithms. Another important development in ImageNet competitions is the adoption of transfer learning. Using pre-trained models as a starting point for new models significantly reduced training time and the data required to accurately train models [23]. A significant benefit of the competition element of the ILSVRC is that it stimulated competing research groups to attempt to outperform each other. It created an environment of innovation where participants constantly strive to develop improved solutions to the given problems. This made the ImageNet competition a competitive playground where inventive ideas for the classification, localization, and detection problems rapidly succeed one another.

Chapter 3

Related Work

This chapter will define and refine the main research question and sub-questions based on existing literature. The topics of this thesis are closely related to different fields of study, namely object detection methods, specifically for the detection of faces, and data augmentation techniques for image datasets.

3.1 Object Detection

Object detection is a method used in computer vision to identify and locate objects in images and videos. Zou et al. [24] state that object detection is one of the most fundamental and challenging problems in computer vision, with many improvements and advancements made over the years. It is considered the basis for many other computer vision tasks such as instance segmentation, image captioning, and object tracking. It is widely used in many real-world applications including autonomous driving, surveillance, and robotics [24]. The technology has come a long way since it was first developed, with many improvements and advancements made over the years. Object detection first began with simple methods like template matching, where a template image is compared to the input image to find the best match at each pixel location [25]. The evaluation was based on the algorithm's accuracy and computational efficiency, which performed well in ideal conditions but had serious limitations in scalability and robustness. Later, methods like edge detection and blob detection were developed, which allowed for the detection of objects based on their edges or certain features [26]. The methods performed well if they accurately detected boundaries and regions in the images. However, edge and blob detection methods were limited in accurately detecting objects in more complex images containing many objects or an agitated background. In the early 2000s, histogram of oriented gradients (HOG) and scale-invariant feature transform (SIFT) methods were developed, which were evaluated based on their ability to detect and match objects across changes in scale, orientation, and lighting conditions. Dalal et al. [27] showed good performance using HOG and SIFT methods for person detection by reducing false positive rates by more than an order of magnitude relative to in comparison to the earlier methods. The techniques above were extensively employed in the field of object detection, with feature engineering playing a significant role in their functioning. However, with the emergence of deep learning, neural network structures have replaced these traditional methods in object detection.

In recent years, CNNs have become the most commonly used method for object detection. CNNs are a type of deep learning algorithm that is able to learn and detect objects in images by analyzing the pixel data directly. This allows them to achieve much higher accuracy than earlier methods.

However, it should be noted that CNNs need large amounts of training data and computation time to train effectively. In literature, CNN based object detection models are often divided into two categories: a two-stage approach and a one-stage approach. The difference between a two-stage and one-stage detector lies primarily in the use of the Region Proposal Network (RPN) and is visualized in Figure 3.1. Two-stage detectors make use of the RPN to identify and classify anchor boxes and then refine the bounding boxes to better align with the ground truth. In contrast, one-stage detectors perform classification and regression on all anchor boxes without relying on the RPN [28]. As two-stage object detection methods, literature introduces region-based convolutional neural networks (R-CNNs), which are able to identify and locate objects in an image by generating a set of bounding boxes around each object and consecutively running a classifier on these proposed boxes. R-CNNs have significantly improved the accuracy and speed of object detection by introducing the two-stage detection framework that separates the region proposal generation and the object detection tasks [29]. In the first stage, the network generates a set of region proposals, which are potential object locations, using a selective search algorithm. The proposed regions are classified and refined in the second stage using a CNN. One-stage object detection methods discussed in the literature primarily include the algorithms YOLO (You Only Look Once) and SSD (Single Shot Detector), which are able to detect objects in real-time and can be used for applications like self-driving cars and security surveillance. These model types are trained on the full image and directly optimize detection performance by minimizing their loss function taking into account both classification and regression losses, making them significantly faster than R-CNNs [30]. Benchmarks such as the PASCAL VOC dataset and the MS COCO dataset evaluate the performance of object detection models in terms of the mean average precision (mAP).

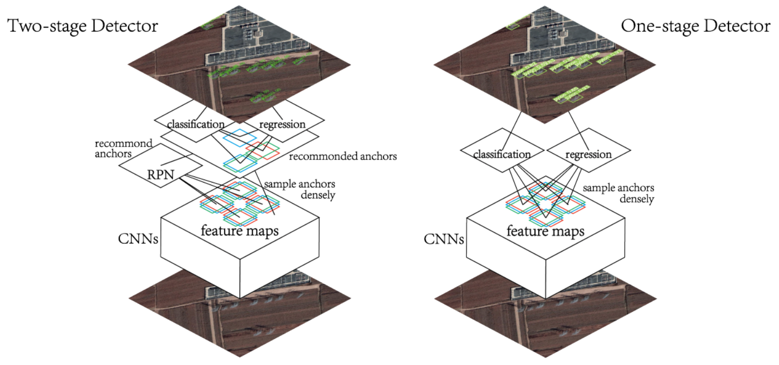


Figure 3.1: *Basic Structure of Two-stage and One-stage Object Detector [28]*

3.1.1 Two Stage Object Detection

Two-stage object detection methods first generate region proposals and then classify these proposals and refine the bounding box coordinates to fit the object better. The region proposal with convolutional neural network (R-CNN) is the first CNN based two-stage object detection model with an architecture based on three components shown in Figure 3.2 [31]. The first segment uses a selective search algorithm to generate a large number of region proposals for each input image. The second segment follows the AlexNet architecture of a CNN with five convolutional layers and two fully connected layers to extract a fixed-length feature vector from each region proposal. Affine image warping [32] is used to obtain the fixed-sized input from each region proposal. The third component classifies each region proposal into one of the object categories that the detection model is trained to detect using a category-specific linear support vector machine (SVM). To summarize, a region proposal is processed by a series of convolutional layers and max pooling layers to extract features. These features are then fed into a fully connected layer, which performs a linear transformation of the feature vector followed by a softmax activation function to output class probabilities for each object category. The employment of R-CNNs for object detection problems resulted in a significant accuracy increase for bounding box prediction [33], achieving a state-of-the-art mAP of 58.5% and 53.3% for the PASCAL VOC 2007 and 2012 dataset for that time, respectively [33].

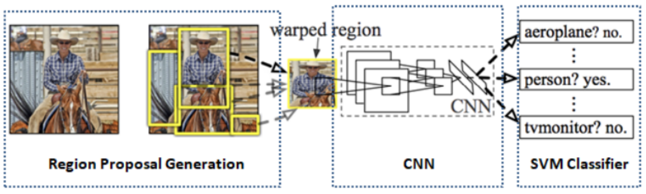


Figure 3.2: *Schematic representation of R-CNN [33]*

However, R-CNN has some serious limitations. The multi-stage approach of feature extraction on each object proposal, network fine-tuning, SVM training and bounding box regression makes inference rather slow and requires cautious hyperparameter tuning to obtain good results. R-CNNs also demand large amounts of training data and computational resources. Girshick et al. [31] modified the R-CNN and proposed Fast R-CNN, a two-stage classifier able to train 9 times faster than R-CNN. Fast R-CNN factorizes the recurrent weight matrix into two smaller matrices using low-rank approximation. This factorization technique reduces the time complexity of Fast R-CNN significantly compared to R-CNN. Girshick et al. found that Fast R-CNN achieves a good result on PASCAL VOC 2012 with a mAP of 68.4% [31].

Both R-CNN and Fast R-CNN depend on the region proposal algorithm for object detection which is a time-consuming computation affecting the overall performance of the network. Ren et al. [34] propose Faster R-CNN, where the region proposal algorithm is replaced with a region proposal network (RPN) which is a fully convolutional network (FCN) taking an image of arbitrary size as input and outputs a set of rectangular candidate object proposals. Faster R-CNN is achieving state-of-the-art object detection accuracy on PASCAL VOC 2007 (73.2% mAP) and 2012 (70.4% mAP) [35].

The multilevel feature map of CNN is used in a different way by Lin et al. [36] to construct Feature Pyramid Network (FPN) resulting in object detection models with superior state-of-the-art performance. FPN uses a bottom-up and top-down architecture, taking feature maps as input and concatenating those to produce the output. RPN is again used to generate region proposals to perform feature extraction. Experimental results on the COCO dataset show a significant detection performance increase for FPN of +3.2 mAP compared to Faster R-CNN [37]. Zhao et al. [38] achieves state-of-the-art detection accuracy on the PASCAL VOC 2007 with 79.2% mAP. On the COCO dataset, a mAP of 62.3% for object detection was achieved by He et al. [39]. According to a comprehensive literature review, this selection of two-stage detection methods includes the most commonly used models. To clarify the differences between all discussed models, Table 3.1 summarizes the most relevant advantages and disadvantages.

Table 3.1: *Advantages and Disadvantages of Two Stage Object Detection Models*

Object Detection Model	Advantages	Disadvantages
R-CNN	Good accuracy Can handle small objects	Slow inference speed High memory use
Fast R-CNN	Faster inference than R-CNN Can handle small objects	Requires selective search
Faster R-CNN	Even faster than Fast R-CNN Handles small and multiple objects	Computationally expensive during training
FPN	Strong performance for different scales	Computationally expensive during training

3.1.2 One Stage Object Detection

In one-stage object detection models, the input is fed through the neural network exactly once, making it possible to predict bounding boxes and class probabilities for all objects in an image in one shot. You Only Look Once (YOLO) is a single-stage model that uses a simple CNN to predict both class probabilities and bounding boxes directly from the input image [30]. An input image is divided into a fixed number of grids that predict a fixed number of bounding boxes with a confidence score, calculated by the difference between the predicted and ground truth bounding box, also known as the IoU. The remaining predictions with an IoU above the threshold value are further explored. For object detection, the YOLO model was first published in 2016 by Redmon et al. in [30]. Subsequent to its initial release, the model has undergone multiple improvements, with development continuing to date. Redmon presents a mAP of 63.4% on the PASCAL VOC 2012 dataset for YOLO and a mAP of 44.0% and 57.9% on the COCO dataset for YOLOv2 and YOLOv3, respectively. YOLOv4 achieved a mAP of 43.5% on the COCO dataset, a state-of-the-art result for that time [40]. Another improvement of the YOLO detector is YOLO9000 [41], a real-time object detector capable of detecting over 9000 object categories by jointly optimizing detection and classification. It achieved a mAP of 76.8% on the PASCAL VOC 2007 dataset, a state-of-the-art result for that time.

The single shot multibox detection (SSD) [42] model also takes a complete image as input and passes it through multiple Conv layers with various size filters. The bounding boxes are predicted using the feature maps from these layers, and the model generates a probability vector corresponding to the confidence level for each object class. Liu et al. [42] achieved a mAP of 74.3% and 68.5% on the PASCAL VOC 2007 and 2012 datasets, respectively. Next, Fu et al. [43] proposed the Deconvolutional Singe Shot Detector (DSSD) to introduce large-scale context in object detectors and improve accuracy for small objects and achieved state-of-the-art performance on both PASCAL VOC 2007 and MS COCO dataset with a mAP of 81.5% and 76.9%, respectively. Last, Lin et al. [44] proposed RetinaNet, a simple one-stage object detector that uses dense sampling of the object locations in an input image. The backbone consists of two sub-networks, the first performing convolutional object classification, and the second performing convolutional bounding box regression. The sub-networks operate as small FCNs connected to the FPN level to be able to share parameters. As a result, mAP of 81.8% and 80.4% were achieved for the PASCAL VOC 2007 and 2012 datasets, respectively. This selection of one-stage detection methods includes the most commonly used models according to a comprehensive literature review. To clarify the differences between all discussed models, Table 3.2 summarizes the most relevant advantages and disadvantages.

Table 3.2: *Advantages and Disadvantages of One Stage Object Detection Models*

Object Detection Model	Advantages	Disadvantages
YOLO	Very fast inference, real-time performance	Difficulty detecting small object
SSD	Fast inference, can handle objects of varying sizes	Difficulty detecting objects at a distance
DSSD	Good accuracy, can handle objects of varying sizes and orientations	Computationally expensive during training
RetinaNet	Good performance on small objects, Effective handling of class imbalance	Computationally expensive during training

In Table 3.3 [33], the performance of the above-mentioned object detection models on the PASCAL VOC and MS COCO datasets are displayed. Again, the model selection is based on the occurrences in a comprehensive literature review.

Table 3.3: *Performance Comparison (mAP) of Object Detection Models on different datasets*

Object Detection Model	PASCAL VOC 2007	PASCAL VOC 2012	MS COCO
Two Stage Models			
R-CNN	58.5%	53.3%	
Fast R-CNN	70.0%	68.4%	
Faster R-CNN	73.2%	70.4%	
FPN	79.2%		62.3%
One Stage Models			
YOLO		63.4%	
YOLOv2			44.0%
YOLOv3			57.9%
YOLOv4			43.5%
YOLO9000	76.7%		
SSD	74.3%	68.5%	
DSSD	81.5%		76.9%
RetinaNet	81.8%	80.4%	

3.2 Data Augmentation

Data augmentation is the process of increasing the amount of data present by making slight alterations in such a way that new data points arise [45]. More training data often makes machine learning algorithms more effective, which makes data augmentation an important step in creating robust models. Traditional transformations that alter and manipulate existing data entail geometric modifications such as cropping, flipping and scaling, while photometric alterations tweak for example the brightness, contrast or saturation of an image [46]. The usefulness of augmentation techniques is highly dependent on the domain and data characteristics, with certain techniques proving more effective in specific domains than others.

3.2.1 Geometric transformations

Geometric transformations alter the geometry of an image by mapping the individual pixel values to new destinations [47]. Transformations such as rotation, cropping and flipping are widely used because of their computationally efficient and easy-to-implement nature. Krizhevsky et al. popularized flipping and cropping in [23] as augmentation techniques for image datasets. Tang et al. [48] proposes a model named PyramidBox with a single-shot architecture. Using augmentation techniques horizontal flipping and random crop, the model achieved a mAP of 94.8% on the WIDER FACE dataset, compared to 92.9% without augmentation. These same augmentation techniques used by Li et al. in [49] resulted in a mAP of 88.8% on the WIDER FACE dataset, compared to 86.3% without augmentation. Furthermore, literature illustrates that random deletion or addition of parts of images can be used to improve a models accuracy [50], robustness [51] or both [52]. In some research, a combination of augmentations is carried out to enrich the training set with transformed original samples, balance the size of the dataset, and avoid overfitting [53].

3.2.2 Photometric transformations

Besides geometric transformations, other popular augmentation techniques to increase the amount of training data are photometric modifications, also known as colour transformations. These kinds of transformations alter the RGB channels of an image by shifting each pixel value *(r, g, b)* to a new pixel value *(r', g', b')* according to a pre-defined heuristic [47]. Popular transformations are enhancing contrast or brightness, white-balancing and sharpening or blurring [54]. Other photometric transformations such as coarse dropout, edge detection, saturation change, inverting and adding noise are proposed by Wang et al. in [55].

Taylor et al. [47] evaluated six geometric and photometric data augmentation methods by implementing them during training of a simple CNN and achieved a significant performance increase generated by the cropping method. Besides research where only geometric or photometric transformations are applied, literature also shows research where these two transformation types are combined. Wu et al. [56] adopted a series of geometric transformations to enrich the training dataset and reduce overfitting. Nair et al. [57] performed two augmentation techniques, namely a random crop combined with a horizontal reflection and varying the intensity of the RGB channels. This prevented overfitting and increased accuracy for the AlexNet. A single-stage joint face detection and alignment method using FPN, single-stage detector, context modelling and cascade regression proposed by Deng et al. [58] achieved a mAP of 56.7% on the WIDER Face And Person Challenge 2019 using data augmentation techniques such as random crop, horizontal flip and colour distortion, against a baseline of 54.8%.

Chapter 4

The Data

On the Vesteda website, potential tenants upload a copy of their identity documents. An available selection from this collection is extracted to serve as a dataset for this research. However, the high variability due to a wide range of upload possibilities and inaccurately uploaded documents ask for data preparation. In addition, the absence of labels in the dataset necessitates annotation to assess the performance of all models.

4.1 Data Preparation

An identity document can be uploaded in three formats: PDF, JPEG, and PNG. Table 4.1 shows the number of uploads of all types. Algorithms having images as input do not accept PDF as input format. It was decided to convert all uploads to PNG format for simplicity and size considerations. Almost all PDF uploads contained multiple pages, such as the front and backside of an identity document. These uploads were separated per page before converting them to PNG format.

Table 4.1: *Number of uploads per image format*

Number of uploads	PDF	JPEG	PNG
21,566	8,312	4,443	8,811

After conversion, the data collection of 30,803 images amounted to more than 100 gigabytes containing images with sizes exceeding 60,000 kilobytes. For processing purposes, it was necessary to carry out a size reduction for all uploads by resizing the width of each image to 400 pixels and altering the height according to the width/height ratio. The resize dimensions were based on considerations of the manageability of the images and the preservation of image quality. This reduced the size of the data collection to around 7 gigabytes. In addition, data exploration made clear that not all uploaded images were identity documents such as passports or identity cards, and not all images of an identity document contained a visible face. Therefore, both image types were manually removed from the collection since the models used in this comparative study only acknowledge images containing labels, i.e., faces. In total, 6,712 images were manually deleted. Figure 4.1 shows the number of files after each preprocessing step.

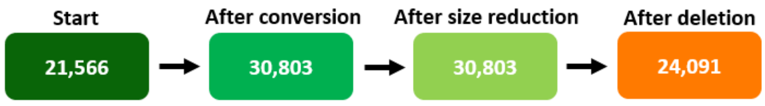


Figure 4.1: *Total number of files after each preprocessing step*

4.2 Data Annotation

After data preparation, the remaining 24,091 images needed to be annotated. Since completing this task manually is very time-consuming, an out-of-the-box YOLOv5 model was used to speed up the annotation process. The YOLOv5 model is primarily employed for real-time detection and is, therefore, excluded from this research, which does not involve any real-time detection. First, a small dataset was created by manually labelling 100 images using LabelImg, a popular graphical image annotation tool created by Tzutalin [59]. LabelImg is part of the Label Studio community, the most flexible open-source data labelling tool for images, text, hypertext, audio, video and time-series data. These 100 images were then used to fine-tune a pre-trained YOLOv5s model [60]. Default settings for image size (640 pixels) and batch size (8 images) were selected, and 30 epochs were carried out. Thereafter, the trained model could be used for inference on all other images to generate output predictions. The annotations of each image were saved in a *.txt* file with the same name as the associated image. Each row in the *.txt* file corresponds to one annotation on that image in the format shown in Table 4.2, where *x_centre* and *y_centre* represent the normalized x and y coordinate of the centre of the bounding box. The output of the YOLOv5 model consists of two folders, one storing all images and the other holding all *.txt* files for each image containing the variables of each annotation per row. Visualizing these predictions with the LabelImg tool showed many inaccuracies; therefore, manual inspection and adjusting were required. The quantity of errors made by the YOLO model is substantial enough to make it unsuitable for the purpose of detecting faces on identity documents.

Table 4.2: *Annotation format of bounding box used by YOLOv5 model*

class	x_centre	y_centre	width	height

Several different types of prediction errors were made by the final model. First, approximately 2,400 images contained faces that were not detected and, therefore, not annotated as faces. Furthermore, around 14,000 images contained annotations that did not point to a face. These bounding boxes enclosed parts of the background, fingers holding down the identity document, blacked out parts of the image, logos or watermarks. Finally, the bounding box size of approximately 6000 annotations was changed since it was too wide or tight around the face. The manual annotation check resulted in a final data collection of 24,091 images containing 43,702 faces.

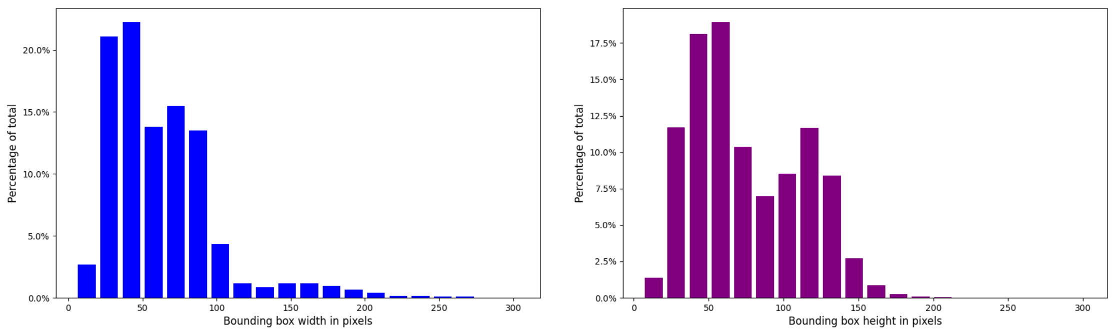
4.2.1 Dataset Structure

The annotation structure of the YOLOv5 model, shown in Table 4.2, differs from those needed by the models used in this comparative study and needs to be changed accordingly. Therefore, a DataFrame containing each annotation from all images in separate rows is created, shown in Table 4.3, where *x_min*, *x_max*, *y_min* and *y_max* represent the minimum and maximum x-coordinate and minimum and maximum y-coordinate of the bounding box, respectively.

Table 4.3: *First five rows of the final data set structure*

	file_name	width	height	x_min	y_min	x_max	y_max	class_name
0	e393d06...978117.png	400	866	249	438	285	482	face
1	e393d06...978117.png	400	866	54	380	120	478	face
2	c437dcd...1730_1.png	400	565	100	310	178	372	face
3	c437dcd...1730_1.png	400	565	112	458	146	483	face
4	c8a54df6...82df_2.png	400	249	34	31	68	80	face

The sizes of the bounding boxes exhibited substantial variation across all images. To illustrate these differences in size, Figure 4.2a and 4.2b present histograms displaying the distributions of bounding box width and height in pixels, respectively. The minimum and maximum values encountered for the width of a bounding box were 5 and 305 pixels, respectively, whereas the minimum and maximum values encountered for the height of a bounding box were 6 and 303 pixels, respectively.



(a) *Distribution bounding box width* **(b)** *Distribution bounding box height*

Figure 4.2: *Size distributions of bounding box width and height in percentages*

A significant amount of the 24,091 images contained multiple faces and thus multiple annotations resulting in a total of 43,702 instances. Figure 4.3 illustrates the percentages of the total count of faces present in each image through a bar plot. A train/validation/test split of 80%/10%/10%, taking into account that the split was not done in such a way that annotations from the same image ended up in multiple sets, resulted in a train, validation and test set of 34,958, 4,374 and 4,370 instances, respectively.

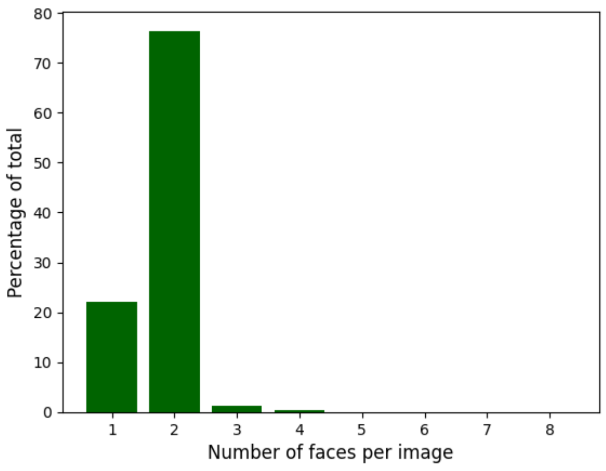


Figure 4.3: *Count of faces present in each image in percentages*

Chapter 5

Methods

Based on an extensive literature review, the implementation of two object detection models, namely Faster R-CNN and RetinaNet, will be carried out. Literature portrays these object detection models as effective models for the purpose of face detection tasks. Ren et al. [35] introduces Faster R-CNN, a faster and more accurate object detection system that uses a region proposal network (RPN) to generate object proposals. RetinaNet, proposed by Lin et al. [44], outperforms other one-stage models for the object detection task and will therefore be implemented in this research. Moreover, the extensive academic research done in Chapter 3 shows there are advantages and disadvantages to both one-stage and two-stage approaches. Typically, two-stage approaches exhibit good accuracy and can handle smaller objects. However, they are computationally demanding and require significant memory use. While one-stage approaches have real-time performance, they can have difficulty detecting small objects. Comparing the two models above, the performance of both one-stage and two-stage approaches can be evaluated for the specific face detection task on identity documents. Both models are equipped with a ResNet-101 backbone and FPN feature extractor. The Feature Pyramid Network (FPN) enhances the quality of feature maps by combining features from different scales. The ResNet-101 backbone uses the weights of a ResNet-101 model as starting point for fine-tuning on a specific object detection task. Recent literature shows the use of this specific backbone for face detection tasks in different environments [61] [62].

In addition to the comparative study, the effectiveness of different data augmentation techniques on improving the performance of detection models will be evaluated. In addition to their ability to generate additional data, the effectiveness of several data augmentation techniques for object detection tasks has been demonstrated in the literature. Geometric transformations such as image flipping, random cropping and rotation improve the robustness of the model to variations in face orientation [63] [64]. Photometric transformations such as contrast and brightness alterations and Gaussian blur can simulate variations of lighting conditions, making detection models robust to changes in illumination [54]. Taylor et al. discuss the use of several geometric and photometric data augmentation methods by implementing them during training [47]. Grounded on an extensive literature review, the data augmentation techniques included in the training process of the aforementioned models are horizontal flip, random crop, colour jittering and a combination of all three. For comparison, the two models are also trained without the use of any data augmentation techniques.

Both detection models deployed in this research include the ResNet-101 and FPN backbone. This architecture will therefore be defined first before further explaining the operations of the two detection models present in the comparative study.

5.1 Backbone: ResNet-101 and Feature Pyramid Network

Within deep learning networks for object detection tasks, the backbone network represents the implemented structure which performs feature extraction. A deep residual network (ResNet) is considered well-suited to serve as the underlying architecture for object detection models because of its excellent performance in various related tasks [63]. ResNet-101 is a convolutional neural network of 101 layers. It utilizes residual connections which enable the training of very deep networks. This architecture is designed to minimize the vanishing gradient problem that often occurs within networks having a large depth. Mathematically, the ResNet-101 network performs a sequence of operations that transform the input image into a feature representation, a concise description of the input image that captures the most important characteristics of the image. First, the input image is passed through a $Conv7 \times 7$ with stride 2, followed by a batch normalization layer and a ReLU activation function. Subsequently, a max pooling layer with 3×3 kernel and stride 2 is applied to reduce the spatial resolution of the feature representation. The output of the max pooling layer, denoted by $f(x)$, is then passed through a series of bottleneck building blocks, shown in Figure 5.1, creating the deep network structure. These residual units are considered the key innovation of a ResNet-101 architecture and reduce the computational cost of the network while improving its accuracy.

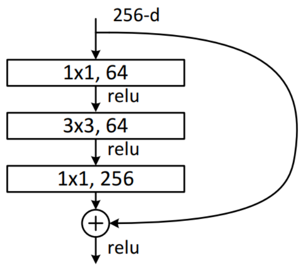


Figure 5.1: *"Bottleneck" building block [63]*

Each bottleneck building block consists of two $Conv1 \times 1$ layers with ReLU activation function separated by a $Conv3 \times 3$ layer, also with ReLU activation function. The $Conv1 \times 1$ layers are responsible for the dimensionality reduction and restoration by decreasing and increasing the number of channels in the input feature maps, leaving the $Conv3 \times 3$ layer as the bottleneck layer having smaller dimensions for input and output. The output of the second $Conv3 \times 3$ layer is added to the input block before passing through another bottleneck building block, also known as the skip connection. After the series of bottleneck residual units, the output of the last residual block is passed through an average pooling layer to reduce the spatial dimensions of each feature map and create the final feature representation of the input image. The schematic representation of the ResNet-101 combined with FPN is shown in Figure 5.2.

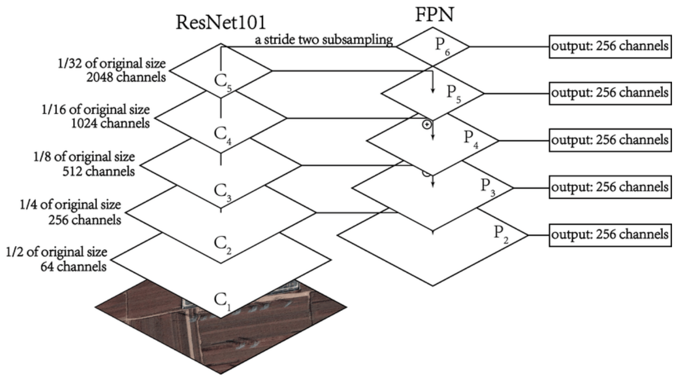


Figure 5.2: *Structure of ResNet-101 backbone and FPN [28]*

The figure shows that ResNet is utilized as the bottom-up pathway and outputs five different feature maps, $\{C_1, ..., C_5\}$, representing features of different scales. Subsequently, the top-down structure of the FPN takes each bottom-up feature map as input to acquire and merge multi-scale features into new feature maps P, to generate a pyramid of feature maps with different spatial resolutions. Bottom-up feature map C_1 is not included in the pyramid because of its large memory requirement. The top-down pathway of the FPN generates these higher-resolution feature maps by upsampling feature maps from higher pyramid levels with lower spatial resolution but capturing higher-level semantic information. These features are then combined with features from the bottom-up pathway using lateral connections. Each lateral connection merges feature maps of the same spatial size from both the bottom-up and top-down pathway [36]. Smaller-sized feature maps (P_6) are spatially coarser, have lower resolutions and therefore capture more contextual information, whereas feature maps covering a smaller receptive field (P_2) are semantically stronger and capture more fine-grained detail and higher-level semantic concepts.

The upsampled feature map is combined with the corresponding bottom-up feature map through element-wise addition after the latter is passed through a $Conv1 \times 1$ layer to reduce its channel dimensions. This operation is repeated for all levels of the FPN pyramid until the finest resolution map is obtained. To start the process, a $Conv1 \times 1$ layer is applied to the feature map of the highest pyramid level (C_5) to produce the feature map with the lowest resolution. Subsequently, the final feature map is obtained after applying a $Conv3 \times 3$ layer to each merged feature map to mitigate the aliasing effect caused by upsampling. The aliasing effect is caused by the loss of high-frequency information during upsampling and can make an image appear less sharp because of distorted edges. The feature dimension, i.e., the number of channels, is kept at a fixed number of 256 because shared classifiers and regressors are used across all levels of the FPN pyramid [36]. Formally, the generation of a feature map P_i can be defined by the formula:

$$P_i = upsample(p_{i+1}) + conv3 \times 3(conv1 \times 1(P_{i-1})),$$

where $upsample()$ represents an upsampling operation of the spatial resolution with factor 2, $Conv1 \times 1$ and $Conv3 \times 3$ are convolutional layers with ReLU activations, respectively. The ResNet-101 backbone and FPN have been pre-trained on a large image dataset, thus already possessing trained weights. For both methods 1 and 2, the models and their weights will undergo further fine-tuning using the identity document dataset.

5.2 Method 1: Faster R-CNN

Faster R-CNN is a two-stage object detector that combines a Region Proposal Network (RPN) to generate object proposals and a Fast R-CNN detector to classify. The model is composed of three main components starting with the ResNet-101 backbone and FPN, followed by the RPN, a fully convolutional network generating high-quality candidate bounding boxes, and a Fast R-CNN network for object classification and bounding box regression. The RPN shares a common set of convolutional layers with the detector network, which reduces the running time of this two-stage detector significantly by accelerating the process of candidate box exploration [35]. Figure 5.3 shows a schematic representation of the Faster R-CNN network.

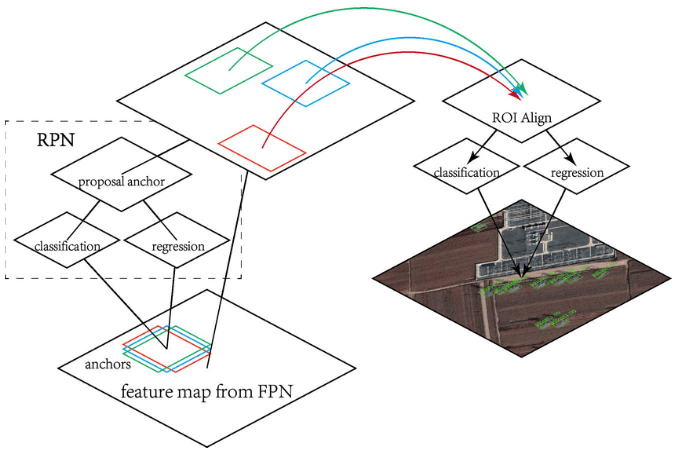


Figure 5.3: *Schematic representation of Faster R-CNN network [28]*

Region Proposal Network Each feature map P_i, generated by ResNet-101 and FPN, is taken as input by the RPN to generate a set of rectangular object proposals p_i. Each object proposal is represented as a bounding box with four coordinates (x_i, y_i, w_i, h_i), where (x_i, y_i) are the coordinates of the top-left corner of the bounding box, and (w_i, h_i) are the width and height of the bounding box, respectively. Additionally, every proposal includes a corresponding confidence score (s_i) indicating the proposal's likelihood of containing that object. The RPN is a fully convolutional network that achieves this output by traversing a small network (the anchor network) over each feature map at different scales and ratios. At each sliding-window location, k region proposals are predicted simultaneously, resulting in $4k$ outputs in the regression layer, the coordinates of k boxes, and $2k$ scores in the classification layer that determine the probability of a proposal being an object or not. The k proposals are characterized by their parameters relative to k reference boxes, referred to as anchors. These anchors are positioned at the sliding window's centre and linked to a particular scale and aspect ratio [35]. To be more precise, at each position of the anchor network, the RPN outputs a set of confidence scores and bounding box offsets, i.e., a representation of the difference between coordinates of the predicted anchor box and nearest ground truth bounding box. The anchor scales are of sizes $32^2, 64^2, 128^2, 256^2$ and 512^2 pixels, and the aspect ratios are 1:1, 2:1 and 1:2, resulting in a total of 15 anchors at each sliding position.

During the training of the RPN, the confidence scores are computed using a binary classifier that predicts whether an anchor box contains an object or not. The bounding box offsets are computed using a regression network that predicts the offsets between each anchor box and the corresponding ground truth bounding box. Furthermore, binary class labels are given to each anchor. A positive label is assigned to anchors with either the highest IoU value with a ground truth box or an anchor having an IoU value higher than 0.7 with any ground truth box. A negative label is given with an IoU lower than 0.3. Anchors that are not classified as positive or negative do not affect the training objective. Although the hyperparameters can be adjusted, this research employs default settings. Following these definitions, the RPN objective function will be minimized, consisting of a classification and regression loss function. These components are trained using the loss functions, which are identical to the classification and regression loss function of the multi-task loss function from Fast R-CNN in [31]:

$$L(z_i, v_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(z_i, z_i^*) + \lambda \frac{1}{N_{reg}} \sum_i z_i^* L_{reg}(v_i, v_i^*), \tag{5.1}$$

where i is the index of an anchor in a batch, z_i represents the predicted probability of the anchor i being an object, and v_i is the vector representation of the parameterized coordinates of the bounding box offsets. z_i^* and v_i^* are the corresponding ground truths of z_i and v_i, respectively. The classification loss L_{cls} is the log loss over the two classes *"object"* and *"not object"* and the regression loss L_{reg} is the robust loss function smooth L1 [35]:

$$L_{cls}(z_i, z_i^*) = -(z_i^* \log(z_i) + (1 - z_i^*) \log(1 - z_i)) \tag{5.2}$$

$$L_{reg}(v_i, v_i^*) = \begin{cases} 0.5(v_i - v_i^*)^2 & \text{if } |v_i - v_i^*| < 1 \\ |v_i - v_i^*| - 0.5 & \text{otherwise,} \end{cases} \tag{5.3}$$

where L_{reg} is only triggered when the anchor is positive ($z_i^* = 1$). The two terms are normalized by N_{cls} and N_{reg}, representing the number of anchor boxes, and weighted with parameter λ to balance the contribution of the two losses [35]. When the confidence scores and bounding box offsets for each anchor box are computed, non-maximum suppression (NMS) is utilized to generate a selection of region proposals based on their confidence score by removing overlapping proposals with a lower confidence score. The IoU threshold for NMS is set to 0.7.

Fast R-CNN The selection of region proposals generated by the RPN is taken as input by the Fast R-CNN detector to output the final set of object detections. First, features are extracted from each region proposal with a Region of Interest (RoI) pooling layer. These extracted features provide a representation of the appearance and spatial characteristics of the object in the region proposal and are used by the model to classify the object and refine its bounding box location. RoI pooling layer divides the region proposal into a fixed number of sub-regions of equal size, and inserts the value in each sub-window corresponding to the output grid cell using max-pooling. This operation results in a fixed-size feature map of each region proposal, regardless of its size or aspect ratio [31]. However, this pooling operation can lead to misalignments between extracted features and the original region proposal, resulting in information loss and reduced accuracy. Therefore, the version of Fast R-CNN used in this research includes a refined version of the RoI pooling layer, namely RoI Align. Since the size of the sub-windows may not align exactly with the size of the pooling grid, bilinear interpolation is used to interpolate the feature values at each grid point. This involves using a weighted average of the four nearest feature points in the feature map to compute the value at the grid point. Bilinear interpolation computes the values of the features at their exact locations corresponding to the sub-regions rather than using the discrete values obtained by max-pooling. RoI Align is able to reduce information loss and improve the accuracy of extracted features in each region proposal [39].

After RoI Align, the extracted features from each region proposal are fed into a series of fully connected layers that predict the confidence scores and bounding box offsets. During training, Fast R-CNN uses a multi-task loss function, as defined in [31], which is also employed as the objective function for the RPN, as mentioned earlier. However, it should be noted that the input for both loss functions is different. The RPN takes all anchor boxes as input, whereas Fast R-CNN receives the region proposals. Finally, the output of Fast R-CNN consists of the final set of bounding box predictions along with their corresponding confidence scores.

5.3 Method 2: RetinaNet

RetinaNet is a one-stage object detection model that extracts features and performs classification and regression directly. The model is composed of three main components starting with the ResNet-101 backbone and FPN, followed by two separate sub-networks for object classification and bounding box regression [28]. Figure 5.4 shows a schematic representation of RetinaNet.

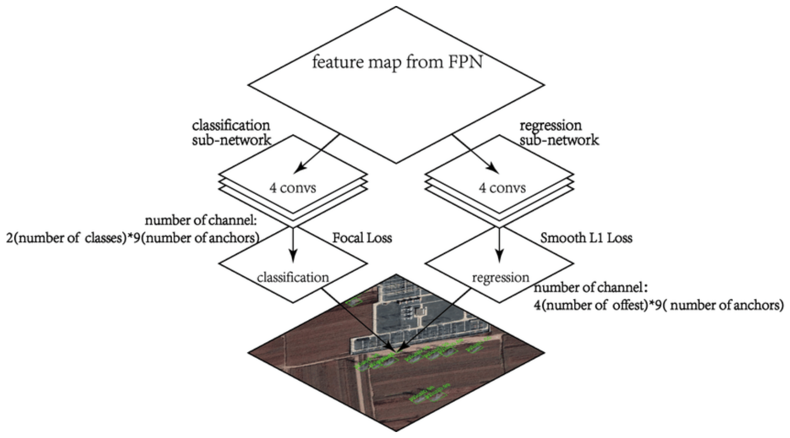

feature map from FPN
classification sub-network
regression sub-network
4 convs
4 convs
number of channel: 2(number of classes)*9(number of anchors)
Focal Loss
Smooth L1 Loss
classification
regression
number of channel: 4(number of offest)*9(number of anchors)

Figure 5.4: *Schematic representation of RetinaNet [28]*

Each feature map P_i, generated by ResNet-101 and FPN, carries different spatial resolutions and is utilized to generate anchor boxes at varying scales. An anchor box is a predefined bounding box placed at different locations and scales in the image. At each spatial location in the feature map, RetinaNet creates a set of k anchor boxes associated with a scale and aspect ratio and parameterizes the centre (x_{ctr}, y_{ctr}), width (w) and height (h) of each anchor box into a 4-tuple (x_{ctr}, y_{ctr}, w, h). Let's denote $A = \{a_1, a_2, ..., a_k\}$ as the set of anchor boxes at a given spatial location. Taking the multi-scale feature maps as input, the classification sub-network predicts the probability of an anchor box containing an object of a certain class, resulting in a set of class probabilities for each anchor box [28]. In order to achieve this, the feature maps $\{P_2, ..., P_6\}$ are passed through the classification sub-network of RetinaNet, consisting of a set of shared convolutional layers followed by a final *Conv*(4×4) layer. This final layer produces a feature map of size ck, where k is the number of anchor boxes and c is the number of object classes. Subsequently, this feature map is then fed into a sigmoid function to produce a c-dimensional vector $z = [z_1, z_2, ..., z_c]$, where z_i is the probability that anchor box a_i contains the object of that specific class [44].

The regression sub-network of RetinaNet also takes the multi-scale feature maps $\{P_2, ..., P_6\}$ as input and is designed to predict the difference between the anchor box coordinates and the corresponding ground truth coordinates of each anchor, also known as the offset. First, the feature maps are fed into a set of shared convolutional layers followed by a final $Conv(4 \times 4)$ layer, which produces a feature map of size $4k$, where k is the number of anchor boxes, and 4 represent the four parameters of each bounding box (i.e., centre coordinates, width and height). Subsequently, the vector of bounding box regression offsets is passed through a linear activation function to produce the final vector denoted by $\mathbf{v} = [v^1, ..., v^k]$, where $v^i = (v_x, v_y, v_w, v_h)$ is the offset of anchor box a_i and represent the centre, width and height of the anchor box, respectively.

$$v^i = \big(v_x, v_y, v_w, v_h\big)^i = \big(\tfrac{x-x_a}{w_a}, \tfrac{y-y_a}{h_a}, \log(\tfrac{w}{w_a}), \log(\tfrac{h}{h_a})\big)^i,$$

where (x_a, y_a, w_a, h_a) are the centre coordinates, width, and height of the i^{th} anchor, and (x, y, w, h), are the corresponding values for the ground-truth bounding box. The model parameters are adjusted in order to minimize the offsets of each anchor.

RetinaNet separately computes the classification and regression loss. For the classification loss, a modified focal loss function is utilized to address the class imbalance issue in object detection datasets, where the number of background samples is significantly higher than that of object samples. Focal loss can be described as a balanced combination of the standard binary cross-entropy loss and a modulating factor that down-weights the contribution of the samples with high confidence scores [44]. This gives more attention to the samples that are harder to detect. The classification loss is defined as:

$$L_{cls}(z_i) = -\alpha_i(1 - z_i)^\gamma \log(z_i), \tag{5.4}$$

where z_i represents the predicted probability of the anchor containing an object. Furthermore, α_i is the balancing parameter addressing the class imbalance, and γ represents the focusing parameter which down-weights the samples with high confidence scores. The regression loss is computed using the smooth L1 loss (Formula 5.3), which is robust to outliers and also used in Faster R-CNN.

$$L(z_i, v_i) = L_{cls}(z_i, z_i^*) + \lambda L_{reg}(v_i, v_i^*) \tag{5.5}$$

The total loss function is a linear combination of the classification and regression losses which are balanced by the weight parameter λ [44].

Once the losses have been calculated, RetinaNet combines the object class probability and the regression offset at each anchor box to obtain a single score that reflects both components. The class probability indicates the likelihood that an object of a particular class is present in that anchor box. At the same time, the regression offset predicts the offset between the predicted box and the ground truth box. These offsets are used to adjust the position and size of the anchor box to more accurately fit the object. To compute the final scores for each anchor box at a specific location, the predicted probability of the object's class is multiplied by the regression offset, effectively combining the information on the class and spatial location. The boxes having the highest score are selected as final detection for that specific location and used to output the coordinates of the bounding box predictions and corresponding confidence scores.

5.4 Data Augmentation Techniques

An extensive literature review has established that data augmentation techniques play a vital role in achieving better performance in object detection tasks. The effectiveness of deep learning models is highly dependent on the availability of large amounts of labelled data. In this research, the dataset consists of identity document images captured under varying conditions, such as different orientations and lighting conditions. Augmenting this data can improve the model's robustness to such variations and enhance its accuracy on new, unseen data [46]. Literature suggests that both geometric and photometric techniques have been used for data augmentation in object detection tasks. Geometric techniques such as horizontal flips and random crops have been popular and have shown good performance. Colour jittering to adjust brightness and contrast has been commonly used for photometric techniques. Additionally, some literature has shown a combination of both types. Hence, in this research, it was decided to apply the following data augmentation techniques both individually and in combination.

Horizontal Flip The horizontal flip is a simple yet effective augmentation technique to increase the diversity of the training set. The original image is flipped from left to right by flipping the image horizontally. Although a horizontally flipped image is not often encountered in camera-captured images of identity documents, it is widely recognized as the most popular technique for augmenting training data and increasing its quantity [35] [65].

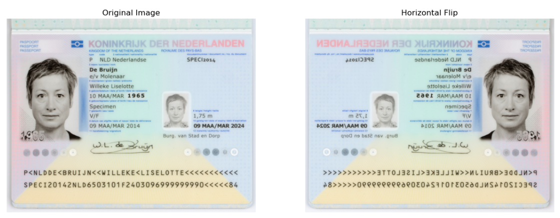


Figure 5.5: *Horizontal flip on identity document copy*

Random Crop This transformation involves randomly selecting a portion of the original image and cropping it to create a new augmented image for training. This transformation takes two arguments: *crop_type* and *crop_size*. The *crop_type* argument can either be *relative* or *absolute*. If *crop_type* is set to *absolute*, *crop_size* should be a 2-tuple specifying the width and height of the cropped region in pixels. If the *crop_type* is set to *relative*, the *crop_size* should be a tuple of two floats specifying the width and height of the cropped region as a fraction of the original image. In this research, a relative crop is done with size $(\frac{1}{3}, \frac{1}{3})$ meaning the crop size is randomly selected as a fraction of the input image size, with both the height and width of the crop ranging from 33% to 100% of the input image size. To clarify hyperparameter choices, their respective values are presented in Table 5.1.

Table 5.1: *Hyperparameter values random crop augmentation*

Hyperparameter	Value
Crop type	Relative
Crop size, width	$\frac{1}{3}$
Crop size, height	$\frac{1}{3}$

The random crop transformation ensures that at least one entire object of interest is contained within the cropped region, guaranteeing the recognizability of the object. Random crop augmentation technique is designed to simulate the occlusion present on camera-captured identity documents. An example of a random crop is presented in Figure 5.6.

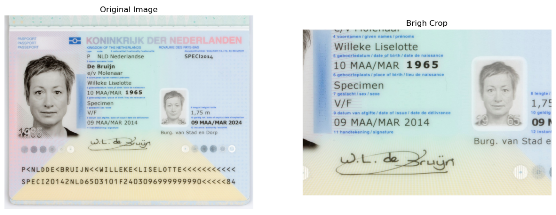


Figure 5.6: *Random crop on identity document copy*

Colour Jittering The goal of colour jittering is to randomly modify the colour or intensity of an image in order to create additional training examples helping the model recognize different lighting conditions and colour variations. In this research, two methods are utilized, specifically colour jittering for adjusting the brightness or contrast. The first technique adjusts the brightness of an image by multiplying the pixel values by a random scaling factor between *intensity_min* and *intensity_max*, which reduces or increases the brightness if the values are below or above 1, respectively. The second technique also chooses the contrast scaling factor randomly between *intensity_min* and *intensity_max* before applying it to the input image. In this research, the *intensity_min* and *intensity_max* are set to $(0.3, 0.8)$ and $(0.2, 1.8)$ for the brightness and contrast technique, respectively. These values are intended to replicate the lighting conditions and colour variations encountered on camera-captured identity documents by imitating bad lighting and heavy sunlight. To clarify hyperparameter choices, their respective values are presented in Table 5.2.

Table 5.2: *Hyperparameter values colour jittering augmentation*

Hyperparameter	Value
Brightness	
Min intensity	0.3
Max intensity	0.8
Contrast	
Min intensity	0.2
Max intensity	1.8

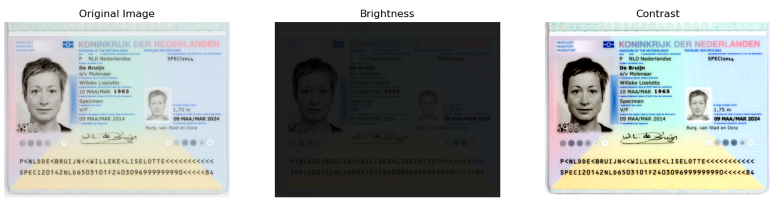


Figure 5.7: *Colour jittering adjusting brightness or contrast on identity document copy*

Combination In this research, a combination of all is additionally carried out besides an individual implementation of the aforementioned data augmentation techniques. All four methods are performed independently with a probability of 0.5.

Chapter 6

Experimental Setup

This research will evaluate the performance of all possible combinations of detection methods and data augmentation techniques previously discussed. Every possibility will be considered a separate detection approach, with a total of 10 possible combinations stated in Table 6.1.

Table 6.1: *Overview of detection methods and data augmentation techniques*

Detection method	Augmentation method
Faster R-CNN	No augmentation
RetinaNet	Horizontal flip
	Random crop
	Colour jittering
	Combination

As stated before, the train/validation/test split adopted in this study is 80%/10%/10%, resulting in 34,958, 4,372, and 4,370 randomly assigned instances present in each set, respectively. The split ensures that annotations of the same image end up in the same part of the sets. All models are developed using Detectron2 [66], a PyTorch-based library. The training process is conducted on a Microsoft Azure NCv3-series virtual machine, leveraging an NVIDIA Tesla V100 GPU with 16 GB of GPU memory and 112 GB of RAM.

6.1 Hyperparameters

The training of the models in both detection methods involves multiple hyperparameters, and a detailed explanation of these parameter values is provided subsequently.

IoU Thresholds The hyperparameter values utilized in this research for the IoU thresholds are the default settings that are commonly used in the literature. In Faster R-CNN, a bounding box is labelled as positive if its IoU with a ground truth box is 0.7 or higher and labelled as negative if its IoU is lower than 0.3. In contrast, for RetinaNet, a bounding box is assigned a positive label if its IoU with a ground truth box is 0.5 or higher, while a negative label is given if its IoU is 0.4 or lower. Bounding boxes that fall between the positive and negative thresholds are ignored.

Loss Parameters In Faster R-CNN, the Cross-Entropy Loss and Smooth L1 Loss are utilized when calculating the classification and regression loss, respectively, for both the RPN and the RoI

Align layer. Weight parameter λ is set to 1 for all experiments. RetinaNet introduces focal loss when calculating the classification loss and sets parameters γ and α to 2 and 0.25, respectively. Again, the weight parameter λ is set to 1. The selection of hyperparameters for all loss functions is based on a comprehensive review of the relevant literature and therefore chosen at their default settings.

Batch Size The batch size parameter represents the number of images the model sees per iteration. A higher batch size generally results in faster training but needs more memory since a higher number of images are loaded into the system at the same time. In this research, the batch size is chosen as high as possible, making optimal use of the present GPU. The choice resulted in a batch size of 16 and 12 images for the Faster R-CNN and RetinaNet, respectively.

Learning Rate The learning rate controls the step size at which the model weights are updated during training, affecting both the training speed and accuracy of the model. A high learning rate can lead to the model diverging and failing to converge to an optimal solution. In contrast, a low learning rate causes the training process to be very slow, resulting in the model getting stuck in a suboptimal solution. Additionally, to prevent this, common practice is to use several warm-up iterations in which the learning rate is gradually increased from a small value to the desired value at the beginning of the training process. Literature shows a good starting point for the learning rate is a value of 0.001. In order to preclude the model from failing to converge, some trial runs of the training process are started with learning rates of 0.001 and 0.0005. The evaluation showed that the model's training process with a lower learning rate was indeed slower, and the model with a higher learning rate did not show any signs of divergence. Therefore it was decided to proceed with a learning rate of 0.001 in the training process. Furthermore, 1000 warm-up iterations were set in which the learning rate linearly increased from a small value of 0.00002 to 0.001. Due to computational constraints, hyperparameter tuning for the learning rate could not be performed.

Number of Epochs After one epoch, a model has seen each training example exactly once and subsequently updates its weights based on the error it made on each example. Training with too few epochs results in a model unable to capture all relevant patterns in the data and can therefore lead to underfitting. Alternatively, too many epochs can lead to overfitting when a model becomes too specialized to the training data and performs poorly on unseen examples. In the aforementioned trial runs, every 2000 iterations, the validation set was evaluated, showing little to no validation loss improvements after 6000 iterations. Therefore, given the size of the training set and batch size, a total of approximately three epochs was carried out. This fast convergence of the model may be attributed to the visual resemblances of images in the dataset, which predominantly contains identity documents generally with shared characteristics.

6.2 Performance Evaluation

The mean average precision (mAP) is the most commonly used evaluation metric for object detection tasks [21]. The mAP is the mean of the average precisions of all object classes. Since this research focuses exclusively on the detection of objects in a single class, specifically faces, the average precision (AP) for this class is equivalent to the mAP. Therefore, utilizing the AP alone is sufficient for the purposes of this study. The AP is computed as the area under the precision-recall curve. In order to understand the concepts of precision and recall, the following components are defined:

True Positive (TP): A correct detection of a ground truth bounding box;

False Positive (FP): An incorrect detection of a nonexistent face or misplaced detection of an existing face;

False Negative (FN): An undetected ground truth bounding box.

It should be noted that the True Negative (TN) metric is discarded in the context of object detection since every image contains an infinite amount of correctly classified negative bounding boxes. The criterion for determining a detection as either correct or incorrect is based on the Intersection over Union (IoU) metric, which is obtained by dividing the shared region between a predicted and ground truth bounding box by their combined region, as illustrated in Figure 6.1.

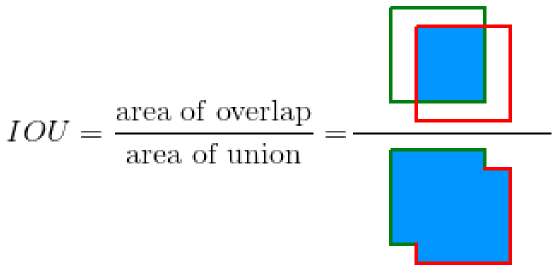


Figure 6.1: *Intersection over Union (IoU) [21]*

To distinguish between correct and incorrect detections, a threshold value t is applied to the Intersection over Union (IoU) metric. The detection is classified as correct if the IoU value is greater than or equal to t. Alternatively, the detection is classified as incorrect if the IoU value is less than t. Based on a comprehensive literature review, an IoU threshold of 0.5 was the most frequently used. Now, the precision P and recall R can be defined as [21]:

$$P = \frac{TP}{TP + FP} = \frac{TP}{all\ detections},$$
$$R = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths},$$

where the precision demonstrates the ability of the model to identify only relevant objects, i.e. faces, and the recall presents the ability of the model to find all possible ground truth boxes. The precision × recall curve represents its trade-off for varying confidence values set by a detector. If the detector wants the number of false positives to be low, the precision will be high, considering only detections with a high confidence level will be included. However, this approach may cause many positives to be missed, resulting in a low recall. On the contrary, if the model lowers the bar for accepting positives, the recall and the number of false positives will increase, resulting in a lower precision. To conclude, a good object detector should find all ground-truth objects while only identifying relevant objects [21]. In other words, with precision staying high as the recall increases, a high area under the precision × recall curve indicates a good object detector.

Since it is rather challenging to accurately measure the area under the precision × recall curve in its zig-zag shape, a 101-point interpolation of the Average Precision (AP) is utilized to summarize the shape of the curve by computing the mean of the highest precision values at 101 uniformly distributed recall levels ranging from 0 to 1, which can be expressed as:

$$AP_{101} = \frac{1}{101} \sum_{R \in \{0, 0.01, \ldots, 0.99, 1\}} P_{interp}(R),$$

where the AP is obtained by utilizing the maximum precision $P_{interp}(R)$ whose recall value is grater than R with $P_{interp}(R) = \max_{\tilde{R}:\tilde{R} \leq R} P(\tilde{R})$. Padilla et al. [21] initially proposed an 11-point interpolation method. However, in these experiments, a 101-point interpolation was utilized for evaluation.

The Average Precision (AP) metric is used to evaluate the model performance on different subsets of the test data based on the IoU threshold between predicted and ground truth bounding boxes. In total, six metrics are calculated:

- **AP**: The overall AP metric, averaged over 10 different IoU thresholds uniformly distributed between 0.5 to 0.95.
- **AP50**: The AP metric at IoU threshold of 0.5.
- **AP75**: The AP metric at IoU threshold of 0.75.
- **APs**: the AP metric for small objects, i.e. boxes with an area smaller than 32^2 pixels.
- **APm**: the AP metric for medium objects, i.e. boxes with area between 32^2 and 96^2 pixels.
- **APl**: the AP metric for large objects, i.e. boxes with an area larger than 96^2 pixels.

Although the above-mentioned metric types are commonly used in literature to evaluate object detection performance [44] [67], the overall AP metric is utilized most frequently. Ren et al. [35] and Girshick et al. [31] use the general AP metric for the evaluation of a Faster R-CNN model and a Fast R-CNN, respectively. Similarly, one-stage object detection models are also evaluated using the general AP metric in [30] and [42] to evaluate a YOLO and SSD detector, respectively. In literature, performance evaluation is also reported using the AP50 metric, which is the most frequently used metric for assessing detection performance on the PASCAL VOC dataset.

Chapter 7

Results

In this section, the various Average Precision metrics for all possible combinations of detection methods and data augmentation techniques will be stated and discussed. In addition, more in-depth insights into the performance differences concerning wrongly classified and missed faces, i.e. false positives and false negatives, will be featured. Last, the different models will be compared based on their performance regarding several image types. The test set consisted of 2,409 images that contained 4,370 ground truth bounding boxes pointing to a face. For each model, a prediction file is generated for the test set, where each prediction consists of an image ID, a category ID, the width and height of the bounding box and the coordinates of the upper-left corner enclosed in a tuple $(x_{min}, y_{min}, b_{width}, b_{height})$, and a confidence score that reflects the certainty of the model regarding the prediction.

7.1 Average Precision

Table 7.1 displays the various Average Precision metrics for all combinations of model and augmentation techniques which are most commonly used in evaluating object detection models according to recent literature. The highest and lowest AP values are denoted in bold and italicized font, respectively. The average precision metrics are calculated with all predictions regardless of their confidence score. Considering the outcomes based on all metric types, the general AP metric is regarded as the main metric in this research. It is the most prevalent metric employed in literature for evaluating the performance of object detection models. RetinaNet without any augmentation illustrates the highest AP score. In contrast, Faster R-CNN with random crop demonstrates the poorest performance on the general AP metric and four of the other metrics. Although RetinaNet combined without any augmentation performed the best on the general AP metric, a one-sample t-test concluded that this result is not significantly better than the second-best model. Interestingly, it is worth noting that the RetinaNet with random crop augmentation exhibits the highest AP score on smaller objects while struggling with larger objects as its APl is lowest. This observation is likely caused by the random crop augmentation technique, making smaller objects easier to predict accurately and larger objects more difficult. However, it should be noted that all models perform considerably lower on the classification of small-sized objects compared to the classification of medium and larger-sized objects. The best-performing model scored an APs of 68.91% contrasted with 82.07% and 82.85% on APm and APl, respectively.

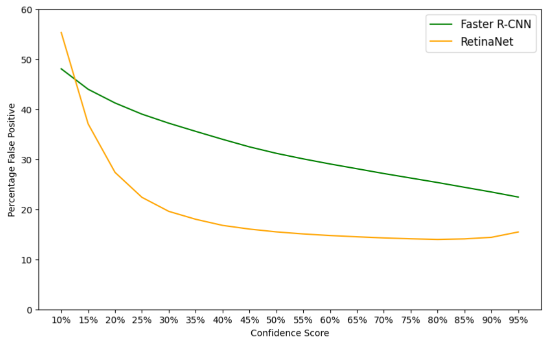
Table 7.1: *Average precision metrics for all model and augmentation combinations*

Model	Augmentation	AP	AP50	AP75	APs	APm	APl
Faster R-CNN	No augmentation	78.96%	98.38%	93.64%	67.44%	81.31%	82.58%
Faster R-CNN	Horizontal flip	75.94%	97.54%	93.29%	*63.90%*	78.61%	79.59%
Faster R-CNN	Random crop	*74.17%*	*97.08%*	*91.22%*	65.75%	*77.40%*	76.35%
Faster R-CNN	Colour jittering	78.93%	98.36%	94.53%	67.50%	81.26%	81.87%
Faster R-CNN	Combination	76.38%	98.50%	93.13%	64.77%	78.78%	79.53%
RetinaNet	No augmentation	**79.54%**	98.67%	**95.10%**	66.88%	**82.07%**	**82.85%**
RetinaNet	Horizontal flip	78.06%	98.60%	93.96%	66.23%	80.36%	80.94%
RetinaNet	Random crop	77.70%	98.65%	92.38%	**68.91%**	79.89%	*75.97%*
RetinaNet	Colour jittering	78.53%	98.63%	93.94%	64.87%	81.37%	82.36%
RetinaNet	Combination	78.04%	**98.77%**	93.83%	65.31%	80.71%	81.83%

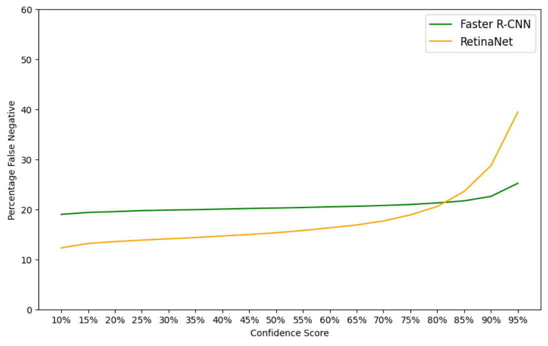
7.2 False Positives and False Negatives

Besides the average precision metric presented above, a detailed examination of the false positive and false negative rate per model and a confidence score is conducted. Given the specific use case of these experiments, it is important to examine the frequency of errors that arise when attempting to locate faces within an image. Such errors can be categorized into two distinct types: firstly, a portion of the image may be mistakenly classified as a face, despite not containing one; and secondly, a face may be entirely overlooked and not identified by the model. These two error types are false positive and false negative predictions, respectively. Vesteda aims to minimize both error types. Avoiding false positives is crucial as it can lead to blurring an important part of the passport. Similarly, false negatives must be avoided as they can result in sensitive parts of the passport being overlooked and left unblurred. The number of false positives is interpreted as the number of predicted bounding boxes mistakenly classified as faces. A bounding box is denoted as a false positive if its IoU with the ground truth of the closest bounding box on that image is below 0.5. This IoU threshold of 0.5 is chosen based on a comprehensive literature review illustrating this value as most commonly used. The total number of predicted bounding boxes a model produces is the number of true positives and false positives together. The number of false negatives is interpreted as the number of missed faces, i.e., faces not detected by the model. As stated earlier, every prediction is made with a certain confidence score, indicating the model's certainty in that prediction. When calculating the false positive and false negative rates, a confidence score of, for example, 60% only considers the predicted bounding boxes with a confidence score of 60% or higher.

Figures 7.1a and 7.1b present the false positive and false negative rates averaged over all data augmentation techniques of the Faster R-CNN and RetinaNet models for several confidence scores, respectively. The Faster R-CNN models demonstrate a gradual reduction in the average false positive rate and a moderate development in the false negative rate as the confidence score increases. In contrast, the RetinaNet models exhibit a swift decrease in the false positive rate with lower confidence values and a rapid increase in the false negative rate as the confidence score reaches higher levels. Overall, the RetinaNet models show a lower percentage of false positives and false negatives compared to Faster R-CNN. Drawing on the insights from Figure 7.1a, it can be concluded that the RetinaNet models often assign a low confidence score to predictions that ultimately turn out to be false positives. Specifically, the false positive rate swiftly decreases as the confidence level increases, indicating that many of RetinaNet's low-confidence predictions are inaccurate. When considering a higher confidence level, these false positives will be effectively



(a) *False positive rate*



(b) *False negative rate*

Figure 7.1: *Percentage of total predictions denoted as false positive 7.1a and false negative rate 7.1b per confidence score with step size 5%*

filtered out. Furthermore, Figure 7.1b suggests that not all true positive predictions of RetinaNet are assigned a high confidence score, as the number of false negatives increases significantly with increasing confidence level. In contrast, the lines representing Faster R-CNN in both figures show a more constant distribution of confidence scores. Together, these observations demonstrate that the confidence scores of RetinaNet's predictions are generally lower and more variable than those of Faster R-CNN. The structural dissimilarities between the two models can account for this observation. Faster R-CNN first generates a set of region proposals using RPN, resulting in a set of possible predictions with high confidence scores, which are then utilized to determine the final set of bounding box predictions. In contrast, RetinaNet divides the image into a fixed number of regions and includes the bounding box with the highest score based on confidence and bounding box offset in each region as part of the ultimate set of bounding box predictions. This differential approach results in Faster R-CNN making a prediction only when highly confident, whereas RetinaNet adopts lower criteria for a final bounding box prediction. It should be noted that the distribution of confidence scores does not influence the performance of the models, as the false positive and false negative rates can be minimized by optimizing the confidence level accordingly.

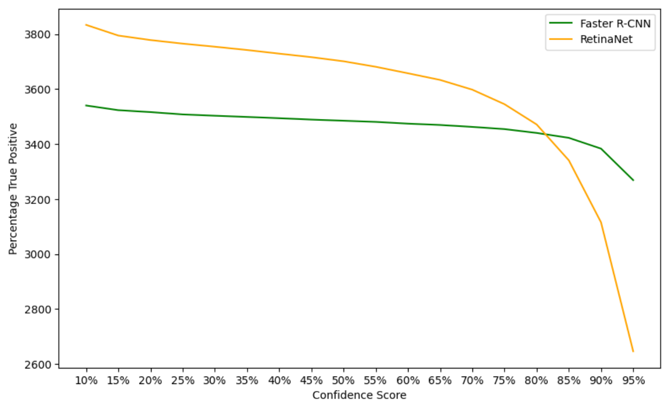


Figure 7.2: *Absolute number of true positives per confidence score*

Figure 7.2 shows the average absolute number of true positives per confidence score for Faster R-CNN and RetinaNet. RetinaNet models, on average, obtain more true positives in comparison to Faster R-CNN models. However, as the confidence score increases, the number of true positives stays constant for Faster R-CNN, whereas RetinaNet shows a rapid decrease once the confidence score reaches higher levels. The figure illustrates that on average RetinaNet models assign a lower confidence score to true positive predictions compared to Faster R-CNN since those models exhibit an earlier decrease in the absolute number of true positives. On average, when making a prediction, Faster R-CNN tends to assign a higher confidence score than RetinaNet. From the false positive and false negative rates can be derived that RetinaNet makes more accurate predictions, albeit with lower confidence scores, in comparison to Faster R-CNN. The aforementioned dissimilarities in the models' structures can also account for this presumption. RetinaNet demonstrates a greater accuracy in detecting objects, i.e. faces, that Faster R-CNN fails to detect.

Taking a closer look at the ratio between false positives and false negatives on average between the two models, a bar plot is presented in Figure 7.3 presenting the percentage of false positives and false negatives stacked per confidence score. Once again, the two models are averaged over all data augmentation techniques applied. Compared to RetinaNet, the Faster R-CNN model displays a notably higher average false positive rate (light green). Furthermore, the average false negative rate for both RetinaNet and Faster R-CNN models (blues) remain relatively constant, with RetinaNet experiencing a sudden escalation once the confidence score increases. From this can be concluded that RetinaNet models, on average, perform better considering the number of wrongly classified and missed faces as performance indicators.

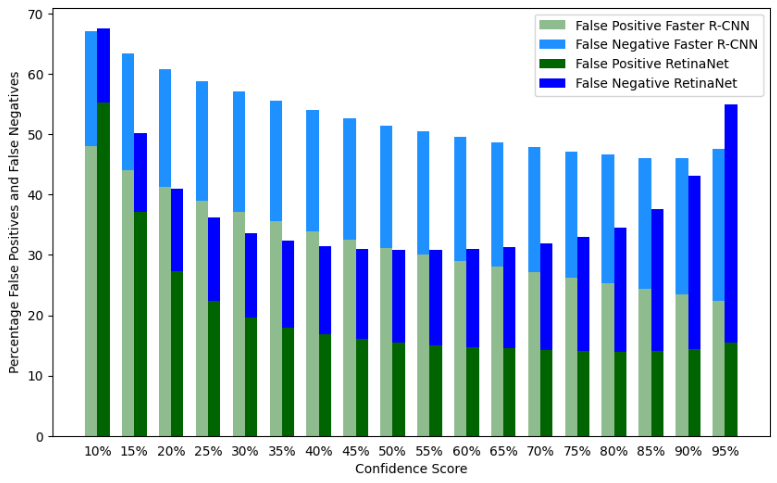


Figure 7.3: *Percentages of false positives and false negatives made by Faster R-CNN and RetinaNet, on average, per confidence score with step size 5%*

As previously mentioned, the evaluation thus far has entailed averaging the performance of various data augmentation techniques utilized in both models. However, notable performance differences can be observed between these techniques. Henceforth, each model's analysis will be conducted separately to enable a comparative analysis of the effects of different augmentation techniques on each model's performance. Subsequent analysis of the false positive and false negative rates will include the confidence score at which the cumulative percentage of the two is minimal. Table 7.2 displays the optimal confidence score for each model. The RetinaNet models exhibit better performance at lower confidence scores than the Faster R-CNN. As stated earlier, this could be attributed to the fact that RetinaNet models tend to generate predictions with lower confidence scores on average, while Faster R-CNN models only provide a prediction when they are highly confident about it.

Table 7.2: *False negative and false positive rate for every model at their relevant optimal confidence score*

Model	Augmentation	False positive	False negative	Confidence score
Faster R-CNN	No augmentation	21.73%	21.03%	85%
Faster R-CNN	Horizontal flip	19.59%	19.7%	85%
Faster R-CNN	Random crop	24.10%	16.68%	90%
Faster R-CNN	Colour jittering	23.77%	23.11%	90%
Faster R-CNN	Combination	30.42%	28.76%	75%
RetinaNet	No augmentation	15.9%	16.0%	50%
RetinaNet	Horizontal flip	13.7%	13.91%	55%
RetinaNet	Random crop	8.11%	8.79%	65%
RetinaNet	Colour jittering	18.2%	18.26%	50%
RetinaNet	Combination	19.58%	19.54%	40%

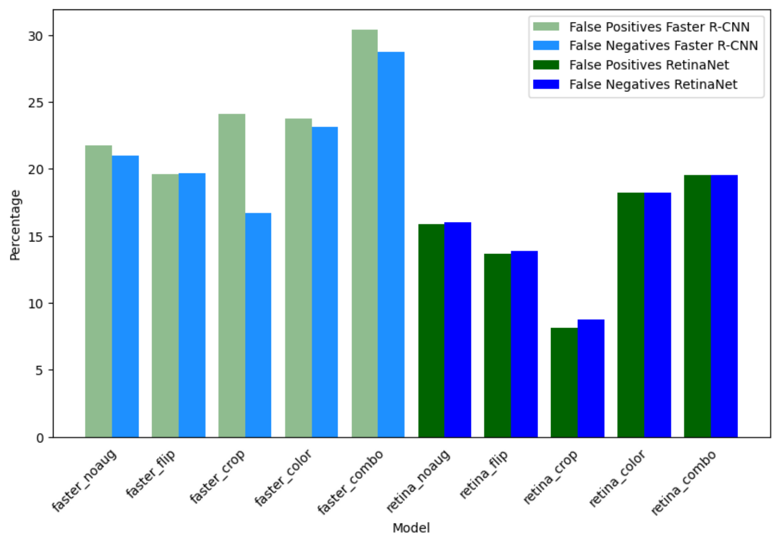


Figure 7.4: *False positive and false negative rate for each model at optimal confidence score*

In Figure 7.4, each model's false positive and false negative rate is shown at its appropriate optimal confidence level. Again, the RetinaNet models have a lower false positive and false negative rate compared to Faster R-CNN, where RetinaNet combined with the crop augmentation resulted in the model with the lowest rates. Additionally, the flip and crop augmentation illustrate lower rates for both model types than the model without any augmentation. Including these augmentation techniques prior to training enabled the models to improve their ability to detect the faces present on the identity documents. Furthermore, both the colour jittering augmentation and the combination of augmentation techniques resulted in the worst performance when considering the false positive and false negative rates as performance indicators. This poor performance may be attributed to the potential confusion caused by applying multiple data augmentation techniques. Confusion may arise from the presence of conflicting augmentation techniques that generate unpredictable outcomes and from the possible loss of critical information due to augmentation techniques.

7.3 Image Type Analysis

To analyze the performance variations of each model based on the image types in the test set, a manual annotation process was conducted on all 2409 images. These images were classified based on ten characteristics, and the resulting distribution of the test set is presented in Table 7.3. For instance, the test set included 432 images that were rotated.

Table 7.3: *Frequency of images labelled with characteristic in test set*

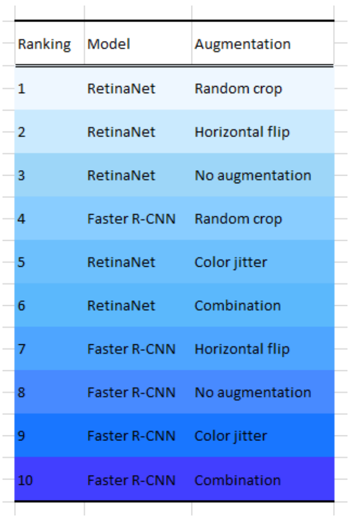
Characteristic	Frequency
rotated	432
small document	143
occlusion	24
bad lighting	85
international document	827
dutch passport	1090
dutch ID	417
dutch drivers license	71
watermark text	464
troubled background	366

Again, the optimal confidence level for each model is utilized for all subsequent analyses. Table 7.4 displays the percentage of images with a specific characteristic that each model correctly predicted at their respective optimal confidence scores. This part of the evaluation defines a prediction as correct if no faces are left undetected, i.e. no false negatives on that image. Only characteristics present in more than 15% of the test set images are included in the table. Each characteristic's highest and lowest percentage are denoted in bold and italicized font, respectively. Faster R-CNN with a combination of augmentation techniques yields the poorest performance on all six characteristics, while RetinaNet with crop augmentation performs the best on five of the six characteristics.

Table 7.4: *Percentage of images with specific characteristic correctly predicted by each model at relevant confidence score*

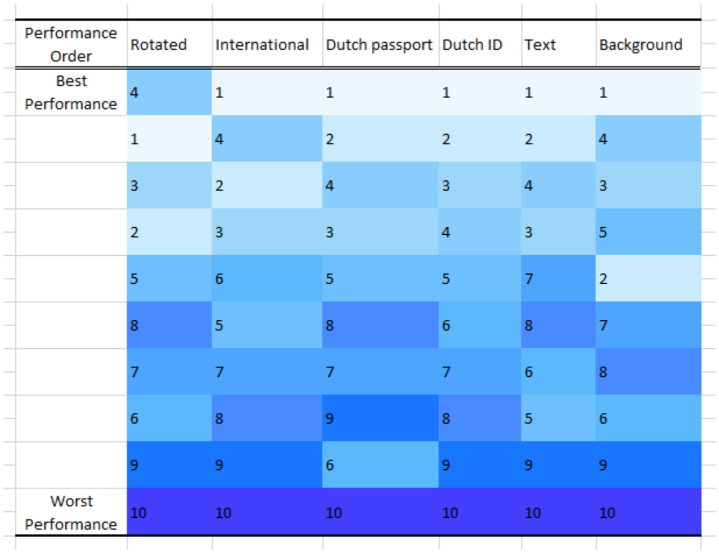
Model	Rotated	International	Dutch passport	Dutch ID	Text	Background
faster_noaug	74.77%	80.05%	63.21%	67.15%	71.98%	65.03%
faster_flip	74.54%	82.83%	62.48%	69.78%	72.84%	68.31%
faster_crop	**86.57%**	86.46%	68.44%	75.3%	74.78%	74.59%
faster_colour	68.98%	78.48%	60.28%	64.99%	68.32%	63.93%
faster_combo	*65.05%*	*77.63%*	*48.62%*	*53.24%*	*64.01%*	*55.19%*
retina_noaug	78.7%	84.4%	67.98%	79.38%	74.57%	72.4%
retina_flip	78.47%	86.34%	70.0%	82.97%	76.29%	71.86%
retina_crop	83.1%	**88.75%**	**84.13%**	**91.13%**	**82.97%**	**84.15%**
retina_colour	77.31%	83.31%	64.22%	74.1%	69.18%	72.4%
retina_combo	72.22%	84.04%	60.09%	72.9%	71.98%	65.03%

The two models that demonstrate the best and worst performance are those that exhibit the lowest and highest false positive and false negative rates displayed in Table 7.2, respectively. It is evident that their performance is comparable to the performance on images featuring a specific characteristic. To determine whether a model performs exceptionally well on a particular characteristic, the performance ranking of each model concerning that characteristic is compared to the performance ranking based on Figure 7.4, and the results are presented in Figure 7.6. It should be noted that the performance ranking is based only on the false negative rate, as a prediction is denoted as correct if no false negatives are present on that image. Figure 7.5 displays the order of performance based on the false negative rate. The models are ranked in descending order, with model 1 (RetinaNet with random crop) performing the best. Figure 7.6 illustrates the performance of all models for each characteristic, also in descending order. The model numbers are presented in each column, revealing that model 4 (Faster R-CNN with random crop) performs the best on rotated images. Notably, Faster R-CNN with random crop performs moderately better on all image types except *Dutch ID* compared to its performance according to Figure 7.5. Additionally, a slightly poorer performance is observed for RetinaNet with a combination of augmentations on rotated images, those with a troubled background, and those containing a Dutch passport. From these observations, the utilization of random crop augmentation positively impacts the model's performance while examining images containing various characteristics.



Ranking	Model	Augmentation
1	RetinaNet	Random crop
2	RetinaNet	Horizontal flip
3	RetinaNet	No augmentation
4	Faster R-CNN	Random crop
5	RetinaNet	Color jitter
6	RetinaNet	Combination
7	Faster R-CNN	Horizontal flip
8	Faster R-CNN	No augmentation
9	Faster R-CNN	Color jitter
10	Faster R-CNN	Combination

Figure 7.5: *Model performance ranked in descending order based on false negative rate*



Performance Order	Rotated	International	Dutch passport	Dutch ID	Text	Background
Best Performance	4	1	1	1	1	1
	1	4	2	2	2	4
	3	2	4	3	4	3
	2	3	3	4	3	5
	5	6	5	5	7	2
	8	5	8	6	8	7
	7	7	7	7	6	8
	6	8	9	8	5	6
	9	9	6	9	9	9
Worst Performance	10	10	10	10	10	10

Figure 7.6: *Model performance based on number of images without missing faces ranked in descending order for every characteristic with numbers corresponding to rank in Figure 7.5*

Chapter 8

Discussion

This study revealed several limitations and areas that warrant further investigation by evaluating and comparing different object detection models and augmentation techniques for detecting faces on identity documents. The subsequent section will address these limitations identified during this research, as well as several potential avenues for future research.

The foremost limitation in this study concerns the data availability, specifically images with complicated characteristics such as rotation, occlusion and bad lighting. The use of advanced deep neural networks such as Faster R-CNN and RetinaNet necessitates a substantial amount of data for training to be able to detect all different types of faces and not miss any hard-to-detect faces. A larger dataset containing more images with complicated characteristics would be beneficial to enhance the model's performance in future research. Furthermore, the manual labelling of the dataset also presents some limitations. Manual annotation is susceptible to errors, with the possibility of missing objects, i.e. faces, or incorrect annotations due to human error during the annotation process. Additionally, some images contain partially blacked-out faces or faces that are not clearly visible, making the annotation challenging due to its dependence on the subjective judgment and interpretation of the annotator. Lastly, Vesteda cannot guarantee the complete absence of duplicate images in the dataset, potentially resulting in identical images in the training, validation, and test set, thereby affecting the performance on the validation and test set.

The comparative study in this research involves two object detection models and three augmentation techniques, selected due to the limited memory size of the Microsoft Azure virtual machine utilised for training. A larger GPU in future research would permit the inclusion of additional detection models and augmentation techniques in a comparative study, as well as accelerate training through the use of larger batch sizes. Although YOLO models are primarily utilised for real-time detection, the thorough literature review conducted in this study demonstrated their strong performance across multiple model versions. As a result, it would be worthwhile to consider integrating them into future comparative research. Default values were employed for many model parameters, such as loss parameters, IoU thresholds for true positive predictions, and the batch size per image representing the number of regions an image is divided into. Future research could provide insightful model performance comparisons by varying these parameters.

All final models were trained until 6000 iterations with a learning rate of 0.001, determined by multiple trials on the training dataset. Incorporating hyperparameter tuning for both the learning rate and the number of iterations would be a beneficial addition in future research to help prevent overfitting. By fine-tuning these hyperparameters, the models can learn to detect objects better and reduce errors such as false positives or negatives. Besides the object detection models, the chosen augmentation techniques possess certain limitations. The random crop and colour jittering augmentation methods were executed using fixed parameters based on literature, which may have significantly influenced the model's performance. Therefore, implementing augmentation techniques with varying parameters would be a useful addition to future research.

The false positive and false negative rates are evaluated using a IoU threshold of 0.5, the most commonly utilised threshold in the literature. However, future research could consider a more comprehensive evaluation of these rates by varying the IoU threshold values. Subsequently, the evaluation treats the importance of both the false positive and false negative rates equally. However, the significance of these rates can vary considerably depending on the specific use case. Therefore, future research could consider a weight-off analysis between the false positive and false negative rates to enable a more nuanced evaluation of the model's performance and facilitate better decision-making in different applications. Furthermore, evaluating the test set based on image type is also subject to certain limitations. Since some characteristics have a relatively small number of occurrences, future research could include a test set with more images and a broader range of characteristics to draw more reliable conclusions. Moreover, the image type analysis defines an accurately predicted as an image without any false negatives. Future research could expand this evaluation by incorporating different definitions of an accurately predicted image, such as one that contains no false positives or false negatives.

Chapter 9

Conclusion

This chapter restates the research questions and presents this study's findings. Furthermore, a summary of the main research findings is provided, and a recommendation is made for Vesteda.

This research focused on the main question: *How accurately can object detection models detect faces on camera-captured identity documents?* To address this question, the process of manual annotation created a dataset containing identity documents, including labels. Subsequently, two object detection models were fine-tuned on that identity document dataset. Their performance was evaluated based on average precision metrics and false positive and false negative rates. Additionally, the two models were combined with various augmentation approaches to augment the available data and ascertain if the augmentation techniques improved the performance of the models. Finally, two sub-questions were formulated to address these comparisons.

SQ 1: *Which object detection models in the literature effectively detect faces in identity documents?*

SQ 2: *How effectively will various data augmentation techniques influence the robustness of a detection model? Will a combination of multiple yield a different effect?*

To address the first sub-question, an extensive literature review was conducted, which revealed that both two-stage and one-stage detection models are commonly used in object detection tasks. Therefore, a two-stage approach, the Faster R-CNN, and a one-stage approach, the RetinaNet, were made for the comparative study. An evaluation of the results showed that the RetinaNet achieved higher average precision on all AP metrics and displayed lower false positive and false negative rates than the Faster R-CNN. The most effective RetinaNet model demonstrated an AP of 79.54%. Nonetheless, this result lacked statistical significance when compared to the second-best model. Furthermore, the mean false positive rate and false negative rate for the RetinaNet models were found to be 15.1% and 15.3%, respectively.

The second sub-question examined the influence of data augmentation techniques on model robustness. Literature review revealed numerous possible augmentation techniques, and this research included three, namely *horizontal flip*, *random crop*, and *colour jittering*. The training dataset was also augmented with a combination of all augmentations to determine if combining all three techniques would yield a different effect. The comprehensive evaluation of the results showed that both horizontal flip and random crop decreased the false positive and false negative rate of both detection models, with random crop resulting in the most significant benefit. Returning to the main question of this research, it has been found that the combination of the RetinaNet detection model and random crop augmentation can effectively detect faces on identity documents with a false positive rate of 8.11% and a false negative rate of 8.97%. These results indicate that this approach has the potential to detect faces on camera-captured identity documents accurately. In conclusion, Vesteda is recommended to use the RetinaNet detection model combined with random crop augmentation for face detection on camera-captured images.

Bibliography

[1] Christina Tikkinen-Piri, Anna Rohunen, and Jouni Markkula. Eu general data protection regulation: Changes and implications for personal data collecting companies. *Computer Law & Security Review*, 34(1):134–153, 2018.

[2] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5525–5533, 2016.

[3] Ronan Sicre, Ahmad Montaser Awal, and Teddy Furon. Identity documents classification as an image classification problem. In *Image Analysis and Processing-ICIAP 2017: 19th International Conference, Catania, Italy, September 11-15, 2017, Proceedings, Part II 19*, pages 602–613. Springer, 2017.

[4] Souhail Bakkali, Muhammad Muzzamil Luqman, Zuheng Ming, and Jean-Christophe Burie. Face detection in camera captured images of identity documents under challenging conditions. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 4, pages 55–60. IEEE, 2019.

[5] Tausif Diwan, G Anirudh, and Jitendra V Tembhurne. Object detection using yolo: challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, pages 1–33, 2022.

[6] Vladimir Nasteski. An overview of the supervised machine learning methods. *Horizons. b*, 4:51–62, 2017.

[7] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9:381–386, 2020.

[8] Sebastian Thrun and Michael L Littman. Reinforcement learning: an introduction. *AI Magazine*, 21(1):103–103, 2000.

[9] FY Osisanwo, JET Akinsola, O Awodele, JO Hinmikaiye, O Olakanmi, J Akinjobi, et al. Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology (IJCTT)*, 48(3):128–138, 2017.

[10] Fatemeh Nargesian, Horst Samulowitz, Udayan Khurana, Elias B Khalil, and Deepak S Turaga. Learning feature engineering for classification. In *Ijcai*, volume 17, pages 2529–2535, 2017.

[11] Kevin Gurney. *An introduction to neural networks.* CRC press, 2018.

[12] Pedro L Fernández-Cabán, Forrest J Masters, and Brian M Phillips. Predicting roof pressures on a low-rise structure from freestream turbulence using artificial neural networks. *Frontiers in Built Environment*, 4:68, 2018.

[13] Rodrigo Oliveira, Ramon CF Araújo, Fabrício JB Barros, Adriano Paranhos Segundo, Ronaldo F Zampolo, Wellington Fonseca, Victor Dmitriev, Fernando S Brasil, et al. A system based on artificial neural networks for automatic classification of hydro-generator stator windings partial discharges. *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*, 16:628–645, 2017.

[14] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[16] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.

[17] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[18] Van Hiep Phung and Eun Joo Rhee. A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. *Applied Sciences*, 9(21):4500, 2019.

[19] Anirudha Ghosh, Abu Sufian, Farhana Sultana, Amlan Chakrabarti, and Debashis De. Fundamental concepts of convolutional neural network. *Recent trends and advances in artificial intelligence and Internet of Things*, pages 519–567, 2020.

[20] Chang Luo, Hanqiao Huang, Yong Wang, and Shiqiang Wang. Utilization of deep convolutional neural networks for remote sensing scenes classification. *Advanced Remote Sensing Technology for Synthetic Aperture Radar Applications, Tsunami Disasters, and Infrastructure*, pages 1–18, 2018.

[21] Rafael Padilla, Sergio L Netto, and Eduardo AB Da Silva. A survey on performance metrics for object-detection algorithms. In *2020 international conference on systems, signals and image processing (IWSSIP)*, pages 237–242. IEEE, 2020.

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[24] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 2023.

[25] Nazanin Sadat Hashemi, Roya Babaie Aghdam, Atieh Sadat Bayat Ghiasi, and Parastoo Fatemi. Template matching advances and applications in image analysis. *arXiv preprint arXiv:1610.07231*, 2016.

[26] Rafał Grycuk, Marcin Gabryel, Marcin Korytkowski, Rafał Scherer, and Sviatoslav Voloshynovskiy. From single image to list of objects based on edge and blob detection. In *Artificial Intelligence and Soft Computing: 13th International Conference, ICAISC 2014, Zakopane, Poland, June 1-5, 2014, Proceedings, Part II 13*, pages 605–615. Springer, 2014.

[27] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[28] Sijia Qiao, Yu Sun, and Haopeng Zhang. Deep learning based electric pylon detection in remote sensing images. *Remote Sensing*, 12(11):1857, 2020.

[29] Wang Zhiqiang and Liu Jun. A review of object detection based on convolutional neural network. In *2017 36th Chinese control conference (CCC)*, pages 11104–11109. IEEE, 2017.

[30] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[31] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[32] George Wolberg. Image morphing: a survey. *The visual computer*, 14(8-9):360–372, 1998.

[33] Farhana Sultana, Abu Sufian, and Paramartha Dutta. A review of object detection models based on convolutional neural network. *Intelligent computing: image processing based applications*, pages 1–16, 2020.

[34] Shaoqing Ren, Kaiming He, Ross Girshick, Xiangyu Zhang, and Jian Sun. Object detection networks on convolutional feature maps. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1476–1481, 2016.

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[36] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[37] Tiancai Wang, Xiangyu Zhang, and Jian Sun. Implicit feature pyramid network for object detection. *arXiv preprint arXiv:2012.13563*, 2020.

[38] Yongqiang Zhao, Rui Han, and Yuan Rao. A new feature pyramid network for object detection. In *2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*, pages 428–431. IEEE, 2019.

[39] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[40] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[41] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[42] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[43] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[44] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[45] Cherry Khosla and Baljit Singh Saini. Enhancing performance of deep learning models with different data augmentation techniques: A survey. In *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, pages 79–85. IEEE, 2020.

[46] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[47] Luke Taylor and Geoff Nitschke. Improving deep learning with generic data augmentation. In *2018 IEEE symposium series on computational intelligence (SSCI)*, pages 1542–1547. IEEE, 2018.

[48] Yuxiang Zhou, Qixiang Ye, Qiang Qiu, Jianbin Jiao, and Xiangyang Li. Pyramidbox: A context-assisted single shot face detector. *IEEE Transactions on Image Processing*, 28(4):2020–2030, 2019.

[49] Jian Li, Yabiao Wang, Changan Wang, Ying Tai, Jianjun Qian, Jian Yang, Chengjie Wang, Jilin Li, and Feiyue Huang. Dsfd: dual shot face detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5060–5069, 2019.

[50] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[51] Dong Yin, Raphael Gontijo Lopes, Jon Shlens, Ekin Dogus Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision. *Advances in Neural Information Processing Systems*, 32, 2019.

[52] Raphael Gontijo Lopes, Dong Yin, Ben Poole, Justin Gilmer, and Ekin D Cubuk. Improving robustness without sacrificing accuracy with patch gaussian augmentation. *arXiv preprint arXiv:1906.02611*, 2019.

[53] Agnieszka Mikołajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem. In *2018 international interdisciplinary PhD workshop (IIPhDW)*, pages 117–122. IEEE, 2018.

[54] Adrian Galdran, Aitor Alvarez-Gila, Maria Ines Meyer, Cristina L Saratxaga, Teresa Araújo, Estibaliz Garrote, Guilherme Aresta, Pedro Costa, Ana Maria Mendonça, and Aurélio Campilho. Data-driven color augmentation techniques for deep skin image analysis. *arXiv preprint arXiv:1703.03702*, 2017.

[55] Xiang Wang, Kai Wang, and Shiguo Lian. A survey on face data augmentation. *arXiv preprint arXiv:1904.11685*, 2019.

[56] Bichen Wu, Forrest Iandola, Peter H Jin, and Kurt Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 129–137, 2017.

[57] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[58] Jiankang Deng, Jia Guo, and Stefanos Zafeiriou. Single-stage joint face detection and alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[59] Tzutalin. Labelimg. Free Software: MIT License, 2015.

[60] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, Kalen Michael, TaoXie, Jiacong Fang, imyhxy, Lorna, (Zeng Yifu), Colin Wong, Abhiram V, Diego Montes, Zhiqiang Wang, Cristi Fati, Jebastin Nadar, Laughing, UnglvKitDe, Victor Sonck, tkianai, yxNONG, Piotr Skalski, Adam Hogan, Dhruv Nair, Max Strobel, and Mrinal Jain. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation, November 2022.

[61] Kaihan Lin, Huimin Zhao, Jujian Lv, Canyao Li, Xiaoyong Liu, Rongjun Chen, and Ruoyan Zhao. Face detection and segmentation based on improved mask r-cnn. *Discrete dynamics in nature and society*, 2020:1–11, 2020.

[62] Samuel WF Earp, Pavit Noinongyao, Justin A Cairns, and Ankush Ganguly. Face detection with feature pyramids and landmarks. *arXiv preprint arXiv:1912.00596*, 2019.

[63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[64] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[65] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.

[66] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[67] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.