SSD-Sface: Single shot multibox detector for small faces

by
Casper Thuis
Student-id: 10341943
August 26, 2018

36 ECTS
January 2017 - August 2018

Supervisor:
Prof. DR. T.E.J. Mensink

Assessor:
Prof. DR. T. Gevers
Abstract

In this thesis we present an approach to adapt the Single Shot multibox Detector (SSD) for face detection. Our experiments are performed on the WIDER dataset which contains a large amount of small faces (faces of 50 pixels or less). The results show that the SSD method performs poorly on the small/hard subset of this dataset. We analyze the influence of increasing the resolution during inference and training time. Building on this analysis we present two additions to the SSD method. The first addition is changing the SSD architecture to an image pyramid architecture. The second addition is creating a selection criteria on each of the different branches of the image pyramid architecture. The results show that increasing the resolution, even during inference, increases the performance for the small/hard subset. By combining resolutions in an image pyramid structure we observe that the performance keeps consistent across different sizes of faces. Finally, the results show that adding a selection criteria on each branch of the image pyramid further increases performance, because the selection criteria negates the competing behaviour of the image pyramid. We conclude that our approach not only increases performance on the small/hard subset of the WIDER dataset but keeps on performing well on the large subset.
Acknowledgements

This thesis would not have been possible without the support and help from others.

Firstly, I would like to thank my thesis supervisor Thomas Mensink at the University of Amsterdam for his help and guidance during my thesis. Aside from guiding me academically he also acted as a coach. Every week he stimulated me to make the most out of this thesis period and to shift your focus on personal activities if things are not going as planned. Moreover, I also learned to focus on the research by keeping track what is important in the research at hand. I thank Thomas, for sharing his knowledge, enthusiasm for research and the reassurance during the project.

Secondly, I would like to thank Fares Alnajar, Roberto Valenti Robert-Jan Bruintjes and Davide Zambrano working at Sightcorp. All of the people at Sightcorp have guided my research through insightful discussions about the material at hand and other related research. Additionally, I was also able to develop myself professionally by reading other research, giving advice to you and having an insight how the industry functions. I thank everybody at Sightcorp for providing a welcome home.

I would also like to thank Alexander van Someren and Ilse van der Linden for being my study partners during my degree and thesis period. By sharing your knowledge you have helped me bring this thesis to a finish.

Finally, I would like to both my brothers Sebastiaan Thuis and Tim Thuis and my parents for supporting me throughout my studies and my thesis.
# Contents

1 Introduction ............................................. 7
   1.1 General Introduction ................................. 7
   1.2 Problem definition ................................. 8
   1.3 Roadmap ........................................... 9

2 Related work ........................................... 11
   2.1 Face detection ....................................... 11
   2.2 Object detection .................................... 12
      2.2.1 Regional box proposal networks ................. 12
      2.2.2 Single shot detectors ............................ 12
      2.2.3 Speed and accuracy trade-off .................... 13
   2.3 Deep learning method for face detection .......... 13
      2.3.1 Regional box proposal networks ................. 13
      2.3.2 Single shot detectors ............................ 14
      2.3.3 Finding tiny faces ............................... 14

3 Methods ............................................... 17
   3.1 SSD network ....................................... 17
      3.1.1 Base network .................................... 17
      3.1.2 SSD layers .................................... 18
      3.1.3 Prediction layers ............................... 18
      3.1.4 Default bounding boxes ......................... 19
   3.2 Loss ............................................... 20
      3.2.1 Localisation loss ............................... 21
      3.2.2 Confidence loss ................................. 22
# 3.3 SSD for small faces

3.3.1 Input resolution

3.3.2 Finding faces at different resolutions

3.3.3 Selection criteria

# 4 Experimental Setup

4.1 Datasets

4.1.1 Wider face dataset

4.2 Evaluation Metrics

4.2.1 Detection results

4.2.2 PR curve explanation

4.2.3 Average precision

4.3 Implementation Details

4.3.1 Parameters for training and inference.

# 5 Results

5.1 Baseline model

5.2 Increasing the resolution for the baseline.

5.2.1 Training on higher resolution

5.3 Image pyramid

5.3.1 Evaluation of the image pyramid

5.3.2 Training on a image pyramid

5.3.3 Selection criteria

5.4 Comparing against state of the art.

# 6 Conclusion

6.1 Future Work

# A Appendices

# B Bibliography
1 Introduction

1.1 General Introduction

When looking at the three dimensional world around us, we humans make sense of the world with ease. Simultaneously, we perceive the objects in the world; we see the scale, shape, rotation, illumination, colour, position and segmentation of the world, and we are able to categorize salient objects e.g., when looking at photos of groups of people we can detect faces from background, recognize people that are familiar and make estimates of their emotions. We can do all this within 13 milliseconds (Potter et al., 2014). Even though psychologists have long studied the visual cortex, it remains unclear how we are able to perform all these tasks with such ease. Adapting to the world around us, we humans quickly learn to recognize objects. One could argue that to get to a basic level intelligence, interaction with the world around us is essential. However, the opposite might also be true, an artificial entity with basic intelligence should be able to detect and recognize objects. This makes the field of object detection a vital step towards creating an artificially intelligent entity.

Whether the statement that an artificial entity with basic intelligence needs to interpret the world around it is correct or not, the field of computer vision is continuously attempting to solve the problem of object recognition and object detection. In the subfield of object detection, deep learning methods such as the convolutional neural network (CNN) are taking the field by storm, due to additions of Krizhevsky et al. (2012). Due to the robustness of CNN’s to a variety of poses, illumination conditions and hardware capabilities, both object detection and face detection have shifted towards CNN methods. In computer vision there are two fields which are closely related, the field of object detection and the field of face detection. This is confirmed by Zhao et al. (2018), who state that recent generic object detection architectures can be applied for specific tasks such as salient detection, pedestrian detection and face detection. As a result these fields have become increasingly intertwined and also the algorithms used to solve these tasks are very similar. For this thesis we will focus on the specific task of face detection.
1.2 Problem definition

Face detection is a fundamental problem in computer vision. The oldest research paper dates back to 1973 (Fischler and Elschlager, 1973). To date it still is a widely researched problem (Hu and Ramanan, 2017). The main reason face detection is a fundamental problem is that faces are crucial for communication. Our faces hold our identity, tell our age and gender and express our emotions. Additionally, face detection plays a crucial role in other face related tasks such as face parsing, face verification, face tagging and face retrieval. Furthermore, other tasks such as age estimation, gender recognition, emotion recognition and pose estimation can benefit from face detection methods by selecting a region of interest and therefore limit the computation (Ranjan et al., 2017). This thesis was written as an internship for Sightcorp¹. Sightcorp is interested in this area, since the face detection can aid their other face related classifiers. In conclusion face detection is a widely studied problem, which is not only researched in academia but also by companies.

One of the reasons why face detection is a widely researched subject is that it has many practical applications. One of those applications is crowd analysis or security analysis; by combining face detection and face tracking one could analyze the movement of people, enabling post analysis of problematic areas or areas for restricted personnel. To date, crowd analysis is performed with Wi-Fi tracking. However, Wi-Fi tracking is flawed. Randomization techniques can be used to prevent tracking. With the help of multiple cameras tracking could possibly provide a solution. The added benefit of tracking with video is that extra demographic analysis can be performed. Businesses such as retail stores are interested in this particular application since face detection could not only improve older more inconsistent methods of people counting, it could also enable age estimation, gender detection and emotion estimation. Which could help optimize the store and its offerings for the right demographic.

Because of the scientific and societal relevance, this study will investigate a method for object detection, particularly Single Shot detector (SSD), and apply this within the face detection domain with specific focus on small faces. Small face to date still represent a problem for face detection since they contain little spatial information. Moreover, context is more important with small faces(50 pixels or less) in comparison to large faces(50 pixels or more) (Hu and Ramanan, 2017). Thus, the approach for detecting small faces is different than the approach for larger faces. One of our approaches will be to increase resolution to help account for the loss in detail when cropping the image resolution to fit the original model. Furthermore, we experiment with increasing the resolution during inference and training to confirm if adding more spatial information increases performance on the wider face dataset. Additionally, we will experiment with an image pyramid architecture to validate whether the SSD method

¹ http://sightcorp.com/
can utilize the predictive power of multiple resolutions. Finally, we introduce a selection criteria to prevent competing behaviour in the image pyramid model. The selection criteria will be applied both during inference, where it will split the predictions of each branch, and during training, where both branches will be optimized for a different face size in the image.

In this thesis the following contributions are made:

- We experiment using a well known object detection framework and test its performance in average precision in face detection.
- We show that increasing image resolution, even without training, can increase performance.
- We show that even the SSD method can benefit from an image pyramid structure while being inherently scale-invariant.
- We experiment with a selection criteria in the image pyramid structure to prevent competing behaviour in the image pyramid structure.

1.3 Roadmap

The outline of this master thesis is as follows: At first, we will discuss background work of non deep learning face detection algorithms, then an overview of popular object detection methods and other state-of-the-art face detection methods. Secondly, a detailed explanation of the SSD method and our adaptations, such as the image pyramid structure, is given. Thirdly, we discuss the dataset properties and the evaluation metrics used in the result and discussion section.
2 Related work

In this section we first will touch upon non deep-learning methods for face detection. Subsequently, we will describe some influential object detection papers. We will finish this chapter with deep learning face detection methods which are heavily influenced by object detection methods.

2.1 Face detection

One of the first papers published on face detection by Fischler and Elschlager (1973); Kanade (1974), which is a indication that face detection is long standing problem for the field of artificial intelligence. Yang et al. (2002) has given us a detailed overview of non deep learning methods. Yang et al. (2002) describes that there are three distinct techniques in non deep-learning face detection methods. The three techniques can be divided into feature-based, template-based and appearance-based.

Feature-based techniques, such as Fischler and Elschlager (1973); Kanade (1974), require the method to locate invariant facial features, such as eyes, mouths and noses, within the image and then use a classifier to determine whether the facial features are in a correct geometrical configuration. The facial features commonly extracted using edge detectors, and are therefore less robust against lighting condition and occlusion. Throughout the years multiple papers Moghaddam and Pentland (1997); Leung et al. (1995); Wiskott et al. (1997); Heisele et al. (2007); Schneiderman and Kanade (2004) have experimented with different methods to extract better features.

Template-based techniques, also known as active appearance models by Cootes et al. (2001), a manually defined template or function is used. In this template correlation between facial features are estimated. Then the template is used as a sliding template over the image to determine location of the face.

In template matching, a standard face pattern (usually frontal) is manually predefined or parameterized by a function. Given an input image, the correlation values with the standard patterns are computed for the face contour, eyes, nose, and mouth independently. The existence of a face is determined based on the correlation values.

The final technique, appearance-based, uses a sliding template approach Sung and Poggio (1998); Romdhani et al. (2001); Viola and Jones (2004), similarly like the template approach. But differs by learning the templates from the training set, instead of the template being defined by humans or parameterized by a function. One of the most used face detection models, which belongs to the appearance-based models, is described by Viola and Jones (2004) and is known
as Viola-Jones algorithm. The Viola-Jones algorithm is a boosting feature based technique that is a fast technique for detecting frontal faces. The algorithm uses Haar features to determine facial features at different scales. By using integral image calculation and a cascading approach, the adaboost algorithm is made more efficient to quickly calculate the likelihood of faces within the image. The Viola-Jones algorithm is known to work well for frontal, non-occluded faces.

2.2 Object detection

2.2.1 Regional box proposal networks

The regional convolutional network (R-CNN) by Girshick et al. (2014) is one of the first successful approaches to combine CNN’s and box proposals. The method is split into a component to localize objects and subsequently a component to classify each box. In the case of RCNN, method uses selective search by Uijlings et al. (2013) to generate proposals from a single image. For each of these proposals features are generated with the help of a CNN. The features are then classified by a SVM to determine the class of the proposal.

Quickly after R-CNN Girshick (2015), an extension was made on their previous work, called fast R-CNN. They made the network faster by first processing the image to generate features by a CNN and then using the proposals from selective search. The proposals from selective search were then applied to the features therefore saving redundant computation on proposals that overlapped. To enable the selection of proposals in the feature map, they introduced the RoI pooling layer. Which selects the features in a region for the pooling is applied. Additionally, they created a single network that was able to classify and regress bounding box proposals.

The most used object detector by Ren et al. (2015) is an extension on the fast R-CNN framework. The extension entailed that proposals were now generated by the network. These proposals were generate by a separate part of the network, called the region proposal network. This enabled the network to be trained end-to-end.

Finally, Dai et al. (2016) made a different adaptation to the R-CNN architecture, called R-FCN. R-FCN also considered one of the more widely used methods, therefore we also mention it. The difference between Faster R-CNN and R-FCN that the cropping of the feature maps is performed later in the classification network. This later cropping saves extra computation and therefore makes the network faster. Furthermore, it uses more position-sensitive score maps in the classification layers to decide whether an object has the right configuration. To help the classifications be more translation invariant.

2.2.2 Single shot detectors

Another approach to combine CNN’s with object detection was made by Redmon et al. (2016) and is called (You Only Look Once) YOLO.
The YOLO is a single feed forward network that process a single image and directly predicts bounding boxes predictions and class confidences from a grid of default bounding boxes. The convolutions generated features and the fully connected layers generate both bounding boxes predictions and class confidences on the final feature map. The property of generating both bounding boxes and class confidence jointly is similar to the R-CNN by Girshick et al. (2014). The difference between R-CNN is that classification and regression is performed on every bounding box directly, whereas R-CNN has a proposal network to filter out predictions. This makes the network more less computationally expensive.

The extension of the YOLO paper is called single shot multibox detector (SSD). The SSD network from Liu et al. (2016), also predicts bounding boxes and classification in one single feed forward but extents it with a hypercolumn approach by Hariharan et al. (2015). The hypercolumn approach includes combines multiple feature maps to detect objects of various sizes. The SSD network is the method we will extent on and therefore will be explained more thoroughly in the method section 3.

2.2.3 Speed and accuracy trade-off

With the brief overview of the most widely used object detectors, selecting an object detection method to experiment with can be challenging. However, Huang et al. (2017) has made a comprehensive study of all the above mentioned object detectors and compared them in a multiple aspects such as, speed, accuracy and memory. This paper has been an foundation for this work in selecting the network architecture. The study concludes that SSD has one of the better trade-offs between speed and accuracy.

2.3 Deep learning method for face detection

With the expansion of deep learning object detection frameworks, the face detection field followed in a similar direction, the shift towards CNNs.

2.3.1 Regional box proposal networks

With Faster R-CNN being one of the most used object detection methods and one of the networks with the highest performance for small faces the method is well suited option for face detection. The paper of Wang et al. (2017a) to adapts the method to face detection, called Face-RCNN. The researches experiment with a center loss function, used in binary classification and experiment with multiscale training and testing to account for the diversity in face sizes in the dataset. Additionally, the paper of Zhang et al. (2018) also to adapts Faster R-CNN for face detection called FDNet1.0. The FDNet has a small adaptation to the architecture of the network in the form off a deformable layer, which helps to detect the small faces
in the dataset. FDNet1.0 also adopts the multi-scale approach. Finally, Wang et al. (2017b) take a slightly different approach. They replace the architecture with the R-FCN architecture and named it Face R-FCN. The main contribution is that the average pooling by the position sensitive operations is replaced weighted average pooling because different facial features may contribute more (eyes are more important than a mouth).

2.3.2 Single shot detectors

Although SSD method is less suited for small objects, it is one of the most efficient method and therefore being used as a method for face detection. The S3FD paper by Zhang et al. (2017b) adapts the network by selecting earlier prediction layers in the convolutional network, adapt anchor sizes and a different sampling method to account for the small faces in the dataset.

As an adaption to the s3fd paper, the authors Zhang et al. (2017a) presented a different approach named FaceBoxes. The method also uses the SSD method as a starting point, however they introduce a smaller network and a adapted base network to account for the smaller faces. Moreover, they also extend on the default bounding box method, which should help the tiling of default bounding boxes over the image.

From the same researches of S3FD the Najibi et al. (2017) adapt the network to be more efficient while still having high performance. The network itself is smaller than the original model and removes some of the convolutional layers and prediction layers to make it more efficient. Furthermore, they introduce two modules, the detection module and the context module, because context as stated by Hu and Ramanan (2017) is crucial for detecting small faces.

Tang et al. (2018) is one of the latest methods for face detection. The researches design a number of different additions to improve the performance on small faces. Firstly, they design the default bounding boxes differently to incorporate more contextual information of the face and the body. On top of this, they fused multiple feature maps together from different scales to join mutually helpful features together. Lastly, the prediction layers takes these joint features in the prediction branch where they propose a context-sensitive prediction module. This module helps incorporate the context information such as, the shoulders and the body to aid in the prediction of the face.

2.3.3 Finding tiny faces

Lastly, Hu and Ramanan (2017) take a more unique approach to face detection and has been of great influence due to their thorough analysis of important aspect relating finding small faces. Their method, which is a hybrid resolution method (HR), uses multiple resolutions to train three separate networks. They use specific templates for different resolutions, to account for the amount of context needed for different face sizes. The networks predictions are then combined,
resulting in a network performs well on both large and small faces. Additionally, the convolutions are shared in all networks to maintain efficiency.
3 Methods

In this chapter we explain our approach to combining object detection frameworks for face detection. We begin by explaining the single shot detection model by Liu et al. (2016) which is a commonly used object detection framework. The loss function used for this task will be discussed separately. We will conclude this chapter by describing the image pyramid structure used by Hu and Ramanan (2017), how it fits in the SSD framework and why we expect it to be beneficial face detection.

3.1 SSD network

The SSD network by Liu et al. (2016) is one of the more commonly used architectures for object detection. The network is a fully convolutional network and can therefore be used for images with any resolution. Two architectures are proposed in the original paper: an architecture for an input resolution of $300 \times 300$ and one for an input resolution of $512 \times 512$ pixels. In this paper our baseline is the $300 \times 300$ model because it is the default model described in the original paper by Liu et al. (2016). Furthermore, other papers also use this network as a baseline Zhang et al. (2017b). One property of the SSD network found by Huang et al. (2017) is that is it more efficient than other detectors and an efficient network was one of the constraints for the Sightcorp application.

The SSD network is called single shot since both object localisation and classification are done within a single feed forward through the network. This is contrast to, for example, the Faster-RCNN network Ren et al. (2015), from which it differs since it does not have a separate regional proposal network. Furthermore, the SSD network combines multiple feature maps with different sizes to generate predictions, similar to Hariharan et al. (2015), to be more scale invariant to objects. These combined predictions from the multiple feature maps produce two outputs, a bounding box offset and a class confidence. The network consists out of three parts, a base network, SSD layers and prediction layers attached to multiple feature maps in the network.

3.1.1 Base network

These first layers are called the base network. The base network consists of stacked convolutions with decreasing size. The purpose of this base network is to provide response maps that enable detections at different sizes. The base network can be seen in figure 3.1 and is represented the convolutions conv1 till conv5. To be consistent with the original paper we use a truncated (fc6 and fc7 removed)
vgg16 base network and initialize those layers with imagenet weights. However, as mentioned by the authors one could replaced the base network by any standard or non standard architecture, e.g inception (Szegedy et al., 2015) or resnet (He et al., 2016).

### 3.1.2 SSD layers

Subsequently to the base network, additional convolutional layers are added: conv6, conv7, conv8, conv9, conv10 and conv11, which are initialized with a truncated normal distribution. These additional convolutions are highlighted as SSD layers in figure 3.1. Similarly to the base network, the decreasing size of the feature helps with generating response maps for various object sizes. These layers however have bigger receptive fields and this helps to detect larger faces.

![SSD network architecture](image)

Figure 3.1: SSD network architecture

### 3.1.3 Prediction layers

The prediction layers are attached to the convolutional base network and SSD layers. For a feature layer of size $m \times m \times c$, where $m$ is the feature map size and $c$ are the number of channels. A convolutional layer is attached with a $3 \times 3 \times r \times (c text{+ off-set coordinates})$ kernel, where $r$ is the number of default bounding boxes and the number of
classes is 2 (face and background). This kernel produces both a face confidence, background confidence and a bounding box offset relative to a default bounding box, which we will touch upon shortly. These prediction layers are attached to multiple points in the convolutional base network and SSD layers, namely conv4_3, conv7_2, conv8_2, conv9_2, conv10_2, conv11_2. The lower layers capture more fine details and are able to capture smaller faces, while higher layers capture more semantically meaningful information and capture larger faces. Therefore, attaching multiple feature layers should help to capture the differently sized faces. All the prediction layers are concatenated at the end of the network, which will result in a single output layer with a fixed number of bounding box predictions.

### 3.1.4 Default bounding boxes

The selective search algorithm Uijlings et al. (2013) has been a vital component in object detection methods in order to obtain region proposals. However, the SSD network has another method for this purpose. The SSD network regresses a grid of default bounding boxes to fit the faces in the dataset. This grid of default bounding boxes is constructed as follows. For each feature map that has a prediction layer attached, we tile bounding boxes on each feature map cell. Which means that every cell of the feature map will have a default bounding box that is centered in the feature cell. The center can be computed as follows,

\[
x_i = i + 0.5 \frac{f_k}{f_k} \quad (3.1)
\]

\[
y_j = j + 0.5 \frac{f_k}{f_k} \quad (3.2)
\]

where \( f_k \) is the length of the size of the square feature map and \( i \) and \( j \) range from 0 till \( f_k \). The original model uses different ratios for their default bounding boxes, as can be seen in figure 3.2. Since faces share the same proportions and are annotated roughly in a one-to-one ratio, we use only one anchor ratio, namely a square. The square ratio is created in two scales. As explained by Liu et al. (2016), the function to compute the default bounding box scale \( s_k \) is,

\[
s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1} (k - 1), k \in [1, m] \quad (3.3)
\]

where \( m \) is the amount of prediction layers, and \( s_{min} = 0.2 \) and \( s_{max} = 0.9 \). The height and width for the two square bounding boxes, \( h_k^1, h_k^2, w_k^1, w_k^2 \) can be computed as follows,

\[
h_k^1, w_k^1 = s_k \quad (3.4)
\]

\[
h_k^2, w_k^2 = \sqrt{s_k s_{k+1}} \quad . \quad (3.5)
\]
3.2 Loss

To optimize the network for both class and bounding box localization, we use a multi-task loss function. Let $x^p_{ij} = \{1, 0\}$ be an indicator ground truth variable for matching the i-th default box with the j-th ground truth box with category $p$. In our case the category of $p$ can be a face or a background class. The matching variable $x^p_{ij}$ is 1 when the IoU(equation 4.1) between the ground truth and default bounding box is higher than 0.5. Furthermore, for each ground truth bounding box, we also match the default box with the highest IoU overlap. The value of $x^p_{ij}$ is thus defined by

$$x^p_{ij} = \begin{cases} j & \text{if IoU} \geq 0.5 \text{ or max IoU} \\ 0 & \text{otherwise} \end{cases} \quad (3.6)$$

Additionally, because of the amount of default bounding boxes, the possibility also exist that more than one bounding box matches the ground truth. The matching of multiple bounding boxes strategy and the selecting of the bounding bounding box with highest IoU overlap, is used to help the learning process with more positive samples to learn on.

The multi-task loss function is defined as,

$$L(x, c, l, g) = L_{\text{conf}}(x, c) + L_{\text{loc}}(x, l, g) \quad (3.7)$$

where the loss consists out of two task losses, the $L_{\text{conf}}(x, c)$, which is the confidence and the bounding box regression loss $L_{\text{loc}}(x, l, g)$. Where $c$ is the class confidence, $l$ the localisation offset prediction, and $g$ is the localisation ground truth.

Figure 3.2: An example of how default bounding boxes are stacked on the image by Liu et al. (2016).
3.2.1 Localisation loss

In the localisation loss, \( L_{\text{loc}} \), a huber loss is used,

\[
L_\delta(d) = \begin{cases} 
\frac{1}{2}d^2 & \text{for } |d| \leq \delta, \\
\delta(|d| - \frac{1}{2}\delta) & \text{otherwise.}
\end{cases} 
\] (3.8)

Where \( d \) is the distance between the predicted localization and the ground truth localization. If we set \( \delta = 1 \), we get the loss function which is known as the smooth \( L_1 \)-loss.

\[
L_{1s}(d) = \begin{cases} 
0.5d^2 & \text{if } |d| \leq 1 \\
|d| - 0.5 & \text{otherwise}
\end{cases} 
\] (3.9)

There are multiple reasons for using the \( L_{1s} \) loss function graphically displayed in figure 3.3. Firstly, the loss function of the \( L_1 \) is not differentiable at 0. Secondly, when \( |d| < 1 \) the loss function has a less steep gradient to better optimize towards the smaller distances. Thirdly, the gradient of the \( L_2 \) becomes too large when the distance is large causing an unstable learning process, whereas the \( L_1 \) loss function has a less hard constraint for points further away from the optimal position. The loss function between the predicted box \( L_{\text{loc}} \) is defined as followed,

\[
L_{\text{loc}}(x, c^p, l_j, g_j) = \frac{1}{N^+} \sum_{i \in \text{Pos}} \sum_{m \in P_{x, c, y, w, h}} x_{ij}^p \cdot L_{1s}(l_m^i - \hat{g}_m^j) 
\] (3.10)

where \( N^+ = \sum_{ij} x_{ij}^p = 1 \), which is a scalar for the amount of positive matches and \( l_i \) is the localization prediction defined as the center off-set and the height and width off-set. The \( d \) in the equation 3.9 is replaced by \( l_m^i - \hat{g}_m^j \). For \( \hat{g}_m^j \) a regression of the prediction center is made relative to its matched default bounding box and defined as followed,

\[
\hat{g}_{cx}^j = (g_{cx}^j - b_{cx}^i) / b_w^i 
\] (3.11)

\[
\hat{g}_{cy}^j = (g_{cy}^j - b_{cy}^i) / b_h^i 
\] (3.12)

\[
\hat{g}_{w}^j = \log\left(\frac{g_w^j}{b_w^i}\right) 
\] (3.13)

\[
\hat{g}_{h}^j = \log\left(\frac{g_h^j}{b_h^i}\right) 
\] (3.14)

The four coordinates of the ground truth are \( g_{cx}^j, g_{cy}^j \), for the center and, \( g_{w}^j, g_{h}^j \), height and width. The \( b_w^i, b_h^i, b_{cx}^i, b_{cy}^i \) respective coordinates of the matched default bounding box. The division of the height and width are used to normalize the width and the height. The log scale are used to balance the differences in scale, this makes the differences in small scale bounding boxes larger and larger differences for large bounding boxes smaller. The same is operations are done on the \( l_m^i \).
3.2.2 Confidence loss

The confidence loss, $L_{conf}$, is a softmax function over the face class and background class denoted with $P$. Because of the large amount of default boxes the negative boxes greatly outnumber the positive bounding boxes. This creates a large class imbalance between background (negative bounding boxes) and faces (positive bounding boxes), which makes the optimization process hard. To counter this issue, hard negative mining is used. Instead of summing over all the negative bounding boxes, the negative bounding boxes are sorted on class confidence and the top $M$ negative bounding boxes are selected. Where the ratio between $M$ and the positive bounding boxes is 3 : 1. The confidence loss is defined as followed,

$$L_{conf}(x,c) = -\frac{1}{N^+} \sum_{i \in \text{Positive}} x_i^p \log(\hat{c}_i^p) - \frac{1}{N^-} \sum_{i \in \text{Negative}} \log(\hat{c}_i^0)$$ (3.15)

where

$$\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$ (3.16)

$$N^- = M.$$

3.3 SSD for small faces

3.3.1 Input resolution

Preliminary results, see section 5.2, indicated that increasing the input resolution used during inference and training have significant influence on the performance of the model, especially on the small faces (50 pixels and less). Since the network is fully convolutional we can increase the input resolution during inference without changing the network. The network does change in the amount of predictions it generates, because the feature layers that are connected to the prediction layers have larger outputs. As a result one can train on an initial resolution and during inference experiment with different resolutions. This eliminates some distortion effects that can occur when re-sizing the image to another resolution. In the result section we will experiment with both and see what effect this has on the performance.

3.3.2 Finding faces at different resolutions

Although the SSD architecture is designed to be scale invariant, the SSD models can still suffer from detecting different scales (Huang et al., 2017). A known approach to create a more scale invariant model is by creating an image pyramid. This approach is commonly used in object detection methods and also in other face detectors, such as Hu and Ramanan (2017); Najibi et al. (2017); Wang et al. (2017a). The idea behind the image pyramid is that different resolutions work
well for different face sizes. The image pyramid is called an image pyramid because the image is processed at different resolutions, called branches, where after it is processed at different resolutions. The branches are processed separately by a shared CNN and afterwards the predictions are combined. This method is effective for the scale problem but comes at an efficiency cost, since the branches do not share computation amongst them.

The image pyramid structure can be seen in figure 3.4. The image pyramid in the figure has two branches but can be extended to multiple branches. Each branch is processed separately by the shared convolutional layers of the network, followed by resolution specific prediction layers which are highlighted by the respond maps in the figure. This will result in specific detections per resolution. These detections per branch will be merged before NMS is performed. The final predictions are the merged predictions after NMS. It is important that NMS is performed on the merged predictions since otherwise the predictions that are in the same branch will be counted as false positives.

We will experiment with two specific configurations, one configuration is during inference, one is during training. In the inference configuration a network is loaded from a single resolution trained model.
The branches share the CNN layers and the response maps/prediction layers thus having the same weights in the branches. Therefore, the only difference between the branches is the resolution. During training only the shared CNN will share the weights for the multiple branches. The prediction layers or responds maps will have separate weights for each resolution. The reason for this is that training with the same prediction layers will likely generate competing behaviour, by detecting the same face multiple times. The loss for each branch is computed similarly as for the original model and then summed together, the loss of the image pyramid is defined as followed,

$$L(x, c, l, g) = \sum_{b=1}^{B} L^b_{\text{conf}}(x, c) + L^b_{\text{loc}}(x, l, g)$$  \hspace{1cm} (3.18)

where B is the set of all branches.

### 3.3.3 Selection criteria

The results in section 5.3 indicate that competing behaviour is a problem. To prevent the network from competing behaviour, we will experiment with a selecting criteria in the loss. For this a particular branch will be focusing on a subset of the predictions, e.g. the first branch will only focus on the faces smaller than 50 pixels and the second branch will only focus on faces larger than 50 pixels. The reason for this selection criteria comes from the analysis in figure 5.6, where we see that models that perform well on faces lower than 50 pixels perform worse on larger faces.

Similar with the other experiment we will apply this selection criteria both during inference and training. For inference the selection criteria is applied only to the predictions of the branches e.g., only a branch will give prediction of faces lower than 50 pixels, while the other branch will give predictions of faces higher than 50 pixels. For training we will optimize the network with the selection criteria. This replaces the $N^+$ in both the confidence loss 3.16 and the localization loss 3.10, to be $N^{<50}$ and $N^{>50}$ for their respective branches.
4 Experimental Setup

In this chapter we describe the experiments we perform to evaluate our models as well as discuss evaluation metrics that are used in evaluation. Moreover, we describe the data sets that are used for training and evaluation and any further implementation details.

4.1 Datasets

For the experiments we use one wider dataset by Yang et al. (2016). The dataset has a training, validation and test split. Due to the time reasons we could not evaluate our result on test set or on a other dataset.

4.1.1 Wider face dataset

The wider face dataset by Yang et al. (2016) is the most frequently used dataset for training deep learning face detection models, Hu and Ramanan (2017); Wang et al. (2017a); Najibi et al. (2017); Wang et al. (2017a); Zhang et al. (2017a); Wang et al. (2017b). The dataset has a training, validation and test split. Due to the time reasons we could not evaluate our result on test set or on a other dataset. The dataset is a relatively difficult dataset as images are taken in an uncontrolled setting, as opposed to other datasets, such as umdfaces (Bansal et al.), PASCAL FACE (Zhu and Ramanan, 2012), FDDDB (Jain and Learned-Miller, 2010), AFW (Yan et al., 2014). The main problem with the above mentioned datasets is that they have low appearance variance, few training images, or are evaluation only datasets. The wider dataset however, contains 32,203 images and 393,703 labelled faces with a high degree of variability in scale, pose and occlusion as depicted in the sample images. wider face dataset is organized based on 61 different events, such as parades or protest. The data has different annotations that are as follows:

- blur: clear, normal blur, heavy blur
- expression: typical expression, exaggerated expression
- illumination: normal illumination, extreme illumination
- occlusion: no occlusion, partial occlusion, heavy occlusion
- pose: typical pose, atypical pose

The dataset also contains difficulties annotation that are assigned by detection rate of EdgeBox (Zitnick and Dollár, 2014) to indicate difference in scale, pose and occlusion. To further analyze our models we included annotation for face sizes from the original resolution.
The original resolution has a width of 1024 and a deviating height per image. The face sizes are divided in the following bins: \([0 − 10, 10 − 50, 50 − 100, 100 − 200, 200 − 400, 400 − 800]\). To give further insights in how the data is distributed over the different properties are shown in 4.1.

4.2 Evaluation Metrics

In this chapter we describe the evaluation metrics used to evaluate our models. The evaluation metrics used are precision recall curve and average precision.

4.2.1 Detection results

The evaluation of detection results requires a metric that determines whether a prediction is correct or not. The Intersection over Union(IoU) is a value used in object detection to measure the relevant predictions. To determine the IoU we need to have the bounding box ground truth \(B_{gt}\) and the bounding box prediction \(B_p\). The IoU is defined as followed,

\[
\text{IoU} = \frac{\text{area}(B_{gt} \cap B_p)}{\text{area}(B_{gt} \cup B_p)}
\]  
(4.1)
Since multiple detections on a single face will be counted as false positives, post-processing of the detections is required. The greedy non-maxima suppression reduces false positives through a number of steps. Firstly, we only consider boxes with a confidence higher than 0.5. We then select the bounding box with the highest confidence and suppress all the bounding boxes that have an IoU bigger than 0.45 with the selected box. The selected bounding box is used as final prediction. This process of selecting the highest confidence bounding box and suppressing the other bounding boxes is repeated until all bounding boxes are either suppressed or considered final prediction. All the remaining positive predictions are sorted by confidence. The highest positive prediction is considered a true positive ($TP$), the other predictions that an IoU $\leq 0.5$ with the ground truth and have less score less are considered false positives ($FP$). The ground truth boxes that have no predictions assigned are considered false negatives ($FN$). True negatives ($TN$) are left out of consideration because true negatives have no influence on the precision and recall.

4.2.2 PR curve explanation

With the definition of the relevant predictions described we can define the metric used to evaluate our models. Precision ($P$) is defined by how much of the prediction are correct, while recall ($R$) is defined by how many predictions are retrieved. Both $P$ and $R$ are defined as followed,

$$P = \frac{TP}{TP + FP} \quad (4.2)$$

$$R = \frac{TP}{TP + FN} \quad (4.3)$$

The precision and recall both show an important aspect of the retrieval performance of the model. Because precision and recall are inversely related, the trade-off between them is important. Moreover, the precision is usually computed at a certain cut-off. The cut-off both influences precision and recall, when the cut-off is higher it increases recall but decreases precision. Precision and recall with cut-off is defined by $P(k)$ and $R(k)$, where $k$ is the cut-off at $k$ bounding boxes. The trade-off between precision and recall can be combined into the precision-recall curve. The curve represents the precision and recall at different threshold values e.g., [0.1, 0.2, ..., 0.9, 1]. At these threshold values the precision and recall is measured. To construct a smooth line the remaining points are interpolated.

4.2.3 Average precision

To further summarize the PR-curve into one metric, the area under the curve (AuC) can be computed. The AuC is the same as the average
precision and can be computed by taking the precision overall values of recall between 0 and 1,

\[ \int_0^1 P(k)dk \quad . \] (4.4)

The integral is an approximation and computed by the sum over precision at all different threshold values multiplied by the change in recall,

\[ \frac{1}{P} \sum_{n=1}^{N} P(k) \Delta R(k) \] (4.5)

where N is the total number of images in the dataset, k is the cut-off at k images and delta r is the change between R(k-1) and r(k).

Instead of the average precision we use the interpolated average precision. The interpolated average precision replaces the precision at cut-off k by the maximum precision observed at all cut-offs with higher recall and is defined as followed

\[ \frac{1}{P} \sum_{n=1}^{N} \max_{k>k} P(\hat{k}) \Delta R(k) \quad . \] (4.6)

### 4.3 Implementation Details

In this section we give a description of implementation details used for the experiments.

The methods are implemented with the Tensorflow framework \(^1\).\footnote{https://www.tensorflow.org/} All the models are trained on a titan x1 of 12gb gpu. There are a number of parameters that are used for training that do not change with the experiments and thus will be listed here for completeness.

#### 4.3.1 Parameters for training and inference.

In training our models some parameters do not change between runs, but are used. For reproducibility of our work we report all the parameters used. For all our models we use the standard SGD optimizer with a learning rate of 0.001 and a batch size of 32. The image pyramid structure requires more memory and therefore the batch size is lower to 4. Furthermore, we use a dropout rate of 0.5 for the dropout layers in conv6 and conv7 and added batch normalisation layers in all the layers except for the base network and the prediction layers. Lastly, we use all the data augmentation methods and hyper parameters mentioned in the original SSD paper (Liu et al., 2016). During inference, we use a NMS IoU overlap of 0.45 and only select prediction boxes of 0.5.
5 Results

In this chapter we present the results of several experiments performed with our SSD model. We first describe the baseline and its shortcomings. The approach is to initially evaluate our models on images with different resolutions following training on images with a higher resolutions. We then extend our results with several experiments such as increasing resolution during inference/training and changing the architecture to an image pyramid architecture. Subsequently, we describe an adaptation by selecting different faces for different branches of the image pyramid to counteract competing behaviour within the image pyramid architecture. We conclude this chapter with a comparison to the state of the art.

5.1 Baseline model

In this section, we evaluate the performance of our baseline SSD model, see section 3.1 for details. The baseline, is trained on 300 × 300 square input images, hence the name $t_3$.

In our first experiment, we evaluate $t_3$ on the different levels of difficulty (easy, medium and hard) as provided in the Wider dataset.

![Data distribution](image1)

Figure 5.1: The graph shows that the highest percentage of faces present in the data set have the lowest recall in the model. While bigger faces seem to be doing relatively well 70% and up.

![Recall vs Precision](image2)

Figure 5.2: The pr-curve for the $t_3$ model. The hard subset has a low precision for recall values higher than 0.25. Furthermore the recall does not reach 100% because NMS only takes the predictions of a confidence of 50% and higher.

In Figure 5.1, we show the results with the data distribution and AP. We observe that $t_3$ performance is quite low, with an overall AP
of 0.23, and for the difficulty levels: easy: 0.79, medium: 0.51, hard: 0.25. The distribution values are: easy: 0.24, medium: 0.43, hard: 0.88. The reason that the distribution does not sum up to 1 is because the difficulties are not exclusive to one level. Full PR curves for are given in Figure 5.2. The 100% recall is not reach because NMS only takes predictions of a confidence of 50% and higher into consideration.

Next we evaluate the performance of $t_3$ as function of size. In Figure 5.3 we show that the model has problems with faces that have a height of 50 pixels or lower.

We conclude that our $t_3$ model performs poorly on hard and/or tiny examples, while these occure the most in the dataset. We hypothesize that this could be explained by the image resolution in combination with the prediction layers attachment. The $t_3$ model resizes the image to $300 \times 300$ resolution. As a result fine details that are present in the image are lost in the process. Furthermore, the feature maps used for prediction down sample the image with a stride of 8, 16, 32 etc. Which together with the down sampling of the images contributes to the fact that fine spatial information, or context is lacking as mentioned by Hu and Ramanan (2017). In the next section we will experiment with increasing the resolution during evaluation to validate this hypothesis.

5.2 Increasing the resolution for the baseline.

In this section we evaluate the hypothesis that increasing the resolution could increase performance for finding small faces. Since our model is fully convolutional, we can increase the input resolution at test time without retraining the filter or parameters. We evaluate the performance by reducing the re-size method from $300 \times 300$ to $500 \times 500$, $700 \times 700$ and $1000 \times 1000$. For clarity we introduce the following abbreviations, the abbreviation $t_3$-e5 stands for trained on $300 \times 300$ and evaluated at $500 \times 500$. 

In table 5.1 and figure 5.1 we show the performance for all the different resolutions. We observe that using higher resolutions is always beneficial for the overall performance. We also observe that $t_3$-e5 does not have highest performance overall but does have best performance in the easy/medium levels. As expected the $t_3$-e10 network (trained on 300, evaluated on 1000) works best on the hard
In Figure 5.5 we show the PR-curves for the 3 levels of difficulty. We observe that increasing the resolution for the t3-e7 and t3-e10 models does increase precision in the hard subset. Furthermore, for lower values of recall the precision of the all the models perform the same. However, it also lowers precision more rapidly for higher values of recall with the easy and medium levels. Increasing resolutions creates less robust models at higher values of recall.

Our last Figure 5.6 shows the performance of the models with respect to face size. We observe, similar for the hard subset, that increasing the resolution improves performance for the face sizes < 50 pixels. Additionally, when increasing resolution the performance for larger face sizes progressively drops.

We conclude that using a higher resolution progressively increases the performance for the small and the hard subset of the data. However, the performance for the other easy, medium and larger faces of the dataset seems to progressively decreases. Moreover, when increasing the resolution during inference the precision drops more rapidly. Ideally a model performs well on all face sizes and will be keep higher precision when raising the value of recall. This behaviour

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>Abbr</th>
<th>Easy</th>
<th>Medium</th>
<th>Hard</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>t3</td>
<td>0.79</td>
<td>0.51</td>
<td>0.25</td>
<td>0.23</td>
</tr>
<tr>
<td>500</td>
<td>t3-e5</td>
<td>0.81</td>
<td>0.76</td>
<td>0.39</td>
<td>0.37</td>
</tr>
<tr>
<td>700</td>
<td>t3-e7</td>
<td>0.72</td>
<td>0.73</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>1000</td>
<td>t3-e10</td>
<td>0.57</td>
<td>0.60</td>
<td>0.45</td>
<td>0.42</td>
</tr>
</tbody>
</table>

Table 5.1: Models with their specification and their abbreviation. The models with lower resolution perform better for the easy/medium subset while the models with higher resolutions perform better for the hard subset and overall AP.

Figure 5.4: The different resolutions and their AP on different difficulties. The models with lower resolution perform better for the easy/medium subset while the models with higher resolutions perform better for the hard subset and overall AP.

Figure 5.5: Pr-curves for subset easy, medium and hard for different inference resolutions. The t3-e10 model rapidly drops in precision for the easy subset, indicating that increasing the resolution does increases error for this subset.
might be caused by that increasing the resolutions is effective for earlier prediction layers, yet less effective for later prediction layers. The off-set predicted by the layers, are trained for the original resolution. When increasing the resolution, the off-set might not be aligned any more. The later prediction layers, attached to the SSD layers, have a larger receptive field which might cause the misalignment to give an greater error. We hypothesize that training on the increased resolution could help to fine-tune these layers to be more effective for the larger face sizes.

5.2.1 Training on higher resolution

To evaluate the hypothesis we train the model on the three aforementioned resolutions. An adjustment to the training settings was required. Due to the higher resolution the batch size was reduced to fit the memory. The batch size used is listed in Table 5.2.

<table>
<thead>
<tr>
<th>Train</th>
<th>Eval</th>
<th>batch size</th>
<th>Abbr</th>
<th>Easy</th>
<th>Medium</th>
<th>Hard</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>500</td>
<td>32</td>
<td>t3-e5</td>
<td>0.81</td>
<td>0.76</td>
<td>0.39</td>
<td>0.37</td>
</tr>
<tr>
<td>500</td>
<td>500</td>
<td>16</td>
<td>5</td>
<td>0.86</td>
<td>0.81</td>
<td>0.43</td>
<td>0.40</td>
</tr>
<tr>
<td>300</td>
<td>700</td>
<td>32</td>
<td>t3-e7</td>
<td>0.72</td>
<td>0.73</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>700</td>
<td>700</td>
<td>8</td>
<td>7</td>
<td>0.86</td>
<td>0.84</td>
<td>0.54</td>
<td>0.51</td>
</tr>
<tr>
<td>300</td>
<td>1000</td>
<td>32</td>
<td>t3-e10</td>
<td>0.57</td>
<td>0.60</td>
<td>0.45</td>
<td>0.42</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>4</td>
<td>110</td>
<td>0.75</td>
<td>0.76</td>
<td>0.55</td>
<td>0.52</td>
</tr>
</tbody>
</table>

Table 5.2: Model with their specification and their abbreviation, these abbreviations will be used in the graphs in this section. Training increases performance for all resolutions.

Table 5.2 and figure 5.7 show the performance for the different resolutions and compare them with their inference counterpart. We

Figure 5.6: Different inference resolutions and their AP as function of size. t3-e5, t3-e7 and t3-e10 perform better for faces below 50 pixels, increasing the overall AP. t3 does better for face above 50 pixel.

Figure 5.7: AP comparison between trained models and inference models. Training always increases AP.
observe that training almost always performs better than inference. Similar to the evaluation on higher resolution images, the t10 model is the best performing model in the hard subset.

The figure 5.8 indicates that training on the specific resolution does aid in the robustness of the model, it reduces the number of false positives.

Lastly, in figure 5.9 we observe that training the network on the specific input resolution increases performance for all the face sizes in the dataset. However, when comparing results for training on a larger resolution against training on lower resolution, the lower resolution still performs better on larger face sizes.

Training on a higher resolution confirms our hypothesis that higher resolution works better for smaller faces. Furthermore, it also confirms that training help with the error generated by increasing resolution in the later layers. However, the training on a higher resolution does not perform better on large face sizes. We see that lower resolutions perform best on the larger face sizes e.g. t5 for 50 pixels and up. While larger resolutions perform the best for smaller face sizes, e.g. t10 for 50 pixels and below. By combining both resolutions in a image pyramid architecture we could utilize the predictive power of both models.
5.3 Image pyramid

In this section we will experiment with the image pyramid architecture, as described in section 3.3.2. We will first evaluate the image pyramid during inference followed an evaluation of a trained image pyramid. This section concludes with the evaluation of adding a selection criteria for different branches of the image pyramid.

5.3.1 Evaluation of the image pyramid

We evaluate a number of different models combinations with the $t_3$ as base model. The reason that we take $t_3$ as baseline is that we can fairly compare with the previous results from section 5.2. All the combinations are listed in table 5.3. We experiment with two and three branches image pyramid setups. Similar as before the highest resolution is $1000 \times 1000$.

<table>
<thead>
<tr>
<th>Evaluation sizes</th>
<th>Abbr</th>
<th>Easy</th>
<th>Medium</th>
<th>Hard</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>300, 500</td>
<td>$t_3$-e35</td>
<td>0.83</td>
<td>0.76</td>
<td>0.39</td>
<td>0.37</td>
</tr>
<tr>
<td>300, 700</td>
<td>$t_3$-e37</td>
<td>0.78</td>
<td>0.76</td>
<td>0.46</td>
<td>0.44</td>
</tr>
<tr>
<td>300, 1000</td>
<td>$t_3$-e310</td>
<td>0.71</td>
<td>0.67</td>
<td>0.49</td>
<td>0.46</td>
</tr>
<tr>
<td>700, 1000</td>
<td>$t_3$-e710</td>
<td>0.66</td>
<td>0.68</td>
<td>0.49</td>
<td>0.46</td>
</tr>
<tr>
<td>700, 1000</td>
<td>$t_7$-e710</td>
<td>0.47</td>
<td>0.43</td>
<td>0.29</td>
<td>0.28</td>
</tr>
<tr>
<td>300, 500, 700</td>
<td>$t_3$-e357</td>
<td>0.83</td>
<td>0.76</td>
<td>0.39</td>
<td>0.37</td>
</tr>
<tr>
<td>300, 500, 1000</td>
<td>$t_3$-e3510</td>
<td>0.83</td>
<td>0.76</td>
<td>0.39</td>
<td>0.37</td>
</tr>
<tr>
<td>300, 700, 1000</td>
<td>$t_3$-e3710</td>
<td>0.78</td>
<td>0.75</td>
<td>0.46</td>
<td>0.43</td>
</tr>
</tbody>
</table>

Table 5.3: Model with their specification and their abbreviation, these abbreviations will be used in the graphs in this section. Networks with large difference in resolutions perform better compared to networks with small difference in resolutions. Three branch networks perform equally or worse than their two branch counterparts. Networks with a $500 \times 500$ branch all have the same AP values.

In the table 5.3 and figure 5.10 we show the results of both difficulties and face sizes. We observe that the image pyramid for the models $t_3$-e310 and $t_3$-e710 utilizes the predictive power of both resolutions the best. If we evaluate the performance of $t_3$-e310 and $t_3$-e710 as function of size. We observe that $t_3$-e310 and $t_3$-e35 are better in large face sizes (50 pixels and up) and $t_3$-e710 is better in small face sizes. Which is inline with previous experiments. Furthermore, models that have branches of smaller differences in resolution e.g., $t_3$-e35, $t_3$-e357 perform worse than models that have larger differences in resolution e.g., $t_3$-e37, $t_3$-e310 and $t_3$-e3710. Models that have a branch with a resolution of $500 \times 500$, all have the same performance. Suggesting that the $500 \times 500$, dominates the predictions. Additionally, image pyramids with three branches consistently perform worse than image pyramids with two branches. The PR curves for difficulties are given in Figure 5.11.

We conclude that the image pyramid architecture can be used to combine the predictive power of both resolutions. The results suggest that branches with small different in resolution show competing behaviour by occupying the same scale space. Other indications for this are that the architectures with a $500 \times 500$ branch all have the same performance (dominated by predictions from that branch) and architectures with three branches perform worse than architectures
with two branches. The increased number of predictions in the same scale space will affect the performance of NMS. In line with our previous experiments we now will train the image pyramid configurations which could further improve our models.

5.3.2 Training on a image pyramid

To limit our scope of training image pyramid models, we select a number of configuration from the previous section. We exclude all image pyramids with three branches, since they do not perform better than their two branch counter parts. Moreover, the image pyramids with three branches require too much GPU memory during training. An important difference with the models from the previous section is that the prediction layers are now branch specific, see section 3.3.2 for details.

Figure 5.10: The AP values for difficulties and sizes for the two and three branch networks grouped.

Figure 5.11: Pr-curve for the models listed in table 5.3. The models with a branch of 500 × 500 also give the exact same PR-curve. The t7-e710 model has low precision amongst most values of recall, currently unknown to us.
In Table 5.4, we observe that the PR curve is more precise at higher values of recall than their inference only counterparts.

From these findings, we conclude that training a pyramid increases performance for models that have larger differences in resolutions. A cause for this might be that the branches will have competing behaviour. This competing behaviour might be traced back to where the prediction layers are attached. If the shared CNN needs to generate features for both branches in the same feature map, the gradients from both branches might counteract each other. To prevent this competing behaviour, we propose a selection criteria on the branches.
Figure 5.12: AP comparison as function of size and difficulty between trained image pyramid and inference pyramid. Training on the image pyramid architecture does not always increase the performance. Only for the models t3\textsubscript{310} and t7\textsubscript{10}.

Figure 5.13: Pr-curve for the models listed in table 5.4.
5.3.3 Selection criteria

We experiment with applying a selection criteria during evaluation and training the image pyramid. We suppress all predictions of 50 pixels and less for the lower resolution branch. For the higher resolution branch we suppress all the prediction of 50 pixels and higher. This selection criteria is formulated because the higher resolutions e.g., 700x700 and 1000x1000 seem to be performing well on the faces of 50 pixels and less. While the 300x300 model is performs better on larger face sizes. We select the best performing models from the previous section and included t7-e710 for completeness.

<table>
<thead>
<tr>
<th>Model/AP</th>
<th>Easy</th>
<th>Medium</th>
<th>Hard</th>
<th>Combined</th>
</tr>
</thead>
<tbody>
<tr>
<td>t3-e310</td>
<td>0.71</td>
<td>0.67</td>
<td>0.49</td>
<td>0.46</td>
</tr>
<tr>
<td>t3-e310-hc</td>
<td>0.78</td>
<td>0.77</td>
<td>0.59</td>
<td>0.56</td>
</tr>
<tr>
<td>t310</td>
<td>0.78</td>
<td>0.76</td>
<td>0.58</td>
<td>0.54</td>
</tr>
<tr>
<td>t310-hc</td>
<td>0.83</td>
<td>0.81</td>
<td>0.61</td>
<td>0.58</td>
</tr>
<tr>
<td>t7-e710</td>
<td>0.47</td>
<td>0.43</td>
<td>0.29</td>
<td>0.28</td>
</tr>
<tr>
<td>t7-e710-hc</td>
<td>0.85</td>
<td>0.83</td>
<td>0.65</td>
<td>0.61</td>
</tr>
<tr>
<td>t710</td>
<td>0.79</td>
<td>0.74</td>
<td>0.51</td>
<td>0.48</td>
</tr>
<tr>
<td>t710-hc</td>
<td>0.85</td>
<td>0.84</td>
<td>0.63</td>
<td>0.60</td>
</tr>
</tbody>
</table>

In table 5.5 and A.2 we observe that formulated selection criteria increases the performance for all models both during inference and training. The t7-e710-hc is the better than t710-hc which is unexpected, because training on the resolution previously increased performance. This result might explained by the reduction in batch size, which was lowered from 4 to 2. Furthermore by examining the models by difficulty level and as function of size, all models consistently perform better with selection criteria than their counterparts without.

In figure 5.15 we observe that most models show high precision values for high values of recall, expect for the t7-e710 model that has a low precision recall curve, the reason for this currently unknown.

In conclusion we can state that the selection criteria helps the image pyramid competing behaviour of the network. The inference models e.g., t3-e310-hc and t7-e710-hc performance increase might be explained by that pre-filtering prediction before NMS. Training with selection criteria further increases the performance, indicating that the branches without selection criteria are competing.

In the appendix figure A.1 we show the results for the other attributed, see section 4.1.1. We compare the results with our initial baseline l3. We observe that the models with selection criteria significantly perform better than our baseline on the attributed, normal blur typical expression, normal illumination, extreme illumination, no occlusion and partial occlusion. Additionally, our model under perform on the attributed typical and atypical pose and heavy occlusion.

In conclusion, we can state that the SSD method, although being scale invariant by design, can still benefit from the image pyramid structure. With the selection criteria being a necessary addition to prevent competing behaviour. Furthermore, our model still performs...
Figure 5.14: AP comparison as function of size and difficulty between image pyramids with selection criteria and without selection criteria. The selection criteria increases performance across all face sizes.
worse on faces with typical and atypical poses, and heavy occlusion.

Figure 5.15: Pr-curve for the models listed in table 5.5.

5.4 Comparing against state of the art.

In this section we will our result against the state of the art mentioned in the related work. An important notice that these works have been done in parallel to our own work, this research began beginning 2017 during which most of these methods have been published, with exception of HR. Nonetheless we report the all the papers mentioned in the related work section, with the pyramidbox currently being the number 1 in the official wider submissions.

<table>
<thead>
<tr>
<th>Model/AP</th>
<th>Easy</th>
<th>Medium</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMS-RCNN</td>
<td>0.90</td>
<td>0.87</td>
<td>0.62</td>
</tr>
<tr>
<td>HR</td>
<td>0.93</td>
<td>0.91</td>
<td>0.81</td>
</tr>
<tr>
<td>Ssh</td>
<td>0.93</td>
<td>0.92</td>
<td>0.84</td>
</tr>
<tr>
<td>Face R-CNN</td>
<td>0.94</td>
<td>0.92</td>
<td>0.87</td>
</tr>
<tr>
<td>S3fd</td>
<td>0.94</td>
<td>0.93</td>
<td>0.86</td>
</tr>
<tr>
<td>Face R-FCN</td>
<td>0.95</td>
<td>0.94</td>
<td>0.83</td>
</tr>
<tr>
<td>FDNet 1.0</td>
<td>0.96</td>
<td>0.95</td>
<td>0.88</td>
</tr>
<tr>
<td>Pyramidbox</td>
<td>0.96</td>
<td>0.95</td>
<td>0.89</td>
</tr>
<tr>
<td>t710-hc</td>
<td>0.85</td>
<td>0.84</td>
<td>0.65</td>
</tr>
<tr>
<td>s3fd baseline(our implementation)</td>
<td>0.86</td>
<td>0.82</td>
<td>0.60</td>
</tr>
<tr>
<td>s3fd baseline</td>
<td>0.92</td>
<td>0.89</td>
<td>0.71</td>
</tr>
</tbody>
</table>

In table 5.4 we show the AP for all the difficulties levels. The overall AP is not shown because the all papers do not report that overall AP. We observe that we do not perform the same as the state-of-the-art on all levels of difficulty. We perform approximately 8% less on the easy and medium subset with the lowest performing model HR. On the hard level we under perform a larger margin, to be precise 17%. We do perform better than CMS-RCNN by Zhu et al. (2016) with a small margin on the hard subset.

since our baseline closely resembles the work of Zhang et al. (2017b), we implemented their baseline model. There baseline models is the similar like our baseline, with the only adaptation being the convolutional layers where the prediction layers are attached. In table 5.4 we compare our implementation of the baseline with the s3fd baseline. We observe that the number have large significant
differences. The reason for this difference might have the following causes. Firstly, data preprocessing might be a problem. Although many papers describe their methods of processing, translating this to another framework can be difficult and not produce the same result. Secondly, due to the large numbers of hyper parameters in SSD method. It might be that some parameters are not properly set or not used. In conclusion we observe that the difference between their implementation is large and therefore might have multiple causes.
6 Conclusion

In this thesis we explored the performance of the SSD model for the face detection tasks. The WIDER dataset is a dataset that contains mostly small faces in a variety of conditions relating to occlusion, illumination, blur, expression, pose and scale. We experimented with applying the original model to the face detection task as explained by the authors, increasing the resolution during inference and training, applying an image pyramid architecture both during inference and training and creating a selection criteria for the image pyramid structure. From the result and discussion section we can draw the following conclusions.

The overall performance of the proposed SSD method is relatively poor. The reason for this seems to be partially due to low resolution and partially due to the fact that the original method was designed to capture objects of various aspect ratios instead of objects of various scales. When increasing the resolution we see that performance increases for the more frequent occurring samples in our dataset (hard and faces of $50 < \text{pixels}$). Here we see that inference/training on a resolution of $700 \times 700$ is the best resolution for the WIDER dataset. This resolution captures the largest range of face sizes. Based on the experiments we conclude that one can train on a lower resolution and get a higher performance by increasing the resolution during inference. To keep the performance consistent amongst all samples in the dataset, we observe that the image pyramid with successfully combines multiple resolutions and that two branches architecture performs the most effective. The models are trained on a combined resolution of $300 \times 300$ with $1000 \times 1000$ and $700 \times 700$ with $1000 \times 1000$. The combination of those models performs best on the most small/hard subset of the dataset, while the models with lower resolution still perform better on the easy/larger subset of the dataset. By applying the selection criteria during inference and training we further increase the performance of image pyramid architecture, by utilizing the predictive power of each branch more effectively. By using the selection criteria the performance seems to increase even further on all levels of difficulty compared to training on the original resolution. There are two reasons for this. One is that the NMS has less predictions to suppress and therefore does a better job. Secondly, training the image pyramid with selection criteria seems to prevent competing behaviour of the network. In conclusion we observe that although the SSD framework is scale-invariant it can still benefit from the image pyramid architecture to detect faces of different sizes. The SSD method together with the image pyramid architecture can be adapted to work for the face detection task.
6.1 Future Work

In conclusion of our presented work, we have learned a couple of important lessons. First off, some time was lost implementing different base networks to validate the performance across different base architectures. However, due to memory and time constraints this effort was lost in the process. A more thorough analysis of the memory requirements of the network, the GPU memory constraints and optimization parameters of the different base architectures could have prevented this. Furthermore, resolution and pre-processing seems to have a large influence on the performance of the network. Another important lesson learned from this study is that when a framework is adapted to another domain or dataset, that resolution and preprocessing seem to have a large influence on the performance of the model. A thorough analysis of input resolution and research of preprocessing methods seems to be vital to create a good baseline.

Possible extensions to our work could start by analyzing which layers generate the most detections. A test indicated that the first layer produced the most detections. Although this is correlated with the fact that the dataset contained mostly small faces and the first layer can generate the most predictions; finding the optimal configuration of layer selection, anchor sizes, and input resolution could further increase performance of the baseline and possibly the image pyramid architecture. Other works such as Zhang et al. (2017b); Najibi et al. (2017) has experimented with layer selection and changing anchor sizes respectively to the input resolution, which was shown to increase the performance.

Another possible extension to our work could be incorporating more context for detection of small faces. As stated by Zhu et al. (2016); Hu and Ramanan (2017); Tang et al. (2018) context for small faces seems important for detection of small faces. The adding of context might help with distinguishing faces from background by learning for example that bodies occur close to faces. The proposed solution for this is fusing multiple detection layers with different receptive fields.
A Appendices
Figure A.1: The attributes in the wider dataset. All the models with the selection criteria are listed here including the baseline $t_3$. We observe that our best models significantly perform better than our baseline on the attributed, normal blur typical expression, normal illumination, extreme illumination, no occlusion and partial occlusion. Additionally, our model under perform on the attributed typical and atypical pose and heavy occlusion.
Figure A.2:
B Bibliography


49


Figures

3.1 SSD network architecture 18
3.2 An example of how default bounding boxes are stacked on the image by Liu et al. (2016). 20
3.3 The $L_1$, $L_2$ and the $L_1_s$ loss functions 21
3.4 The image pyramid structure 23

4.1 Wider distribution over different annotations 26
4.2 The IoU overlap graphically displayed by www.pyimagesearch.com 27
4.3 Precision and recall graphically displayed by en.wikipedia.org/wiki/Precision_and_recall 27

5.1 The graph shows that the highest percentage of faces present in the data set have the lowest recall in the model. While bigger faces seem to be doing relatively well 70% and up. 29
5.2 The pr-curve for the $t_3$ model. The hard subset has a low precision for recall values higher than 0.25. Furthermore the recall does not reach 100% because NMS only takes the predictions of a confidence of 50% and higher. 29
5.3 The $t_3$ AP as function of size. The prediction distribution and the data distribution have little overlap. 30
5.4 The different resolutions and their AP on different difficulties. The models with lower resolution perform better for the easy/medium subset while the models with higher resolutions perform better for the hard subset and overall AP. 31
5.5 Pr-curves for subset easy, medium and hard for different inference resolutions. The $t_3$-e10 model rapidly drops in precision for the easy subset, indicating that increasing the resolution does increases error for this subset. 31
5.6 Different inference resolutions and their AP as function of size. $t_3$-e5, $t_3$-e7 and $t_3$-e10 perform better for faces below 50 pixels, increasing the overall AP. $t_3$ does better for face above 50 pixel. 32
5.7 AP comparison between trained models and inference models. Training always increases AP. 32
5.8 PR-Curve comparison between trained models and inference models. Trained models have higher values of precision for higher values of recall. 33
5.9 AP comparison as function of size between trained models and inference models. The lowest resolution still performs the best on larger faces. 33
5.10 The AP values for difficulties and sizes for the two and three branch networks grouped. 35
5.11 Pr-curve for the models listed in table 5.3. The models with a branch of $500 \times 500$ also give the exact same PR-curve. The t7-e710 model has low precision amongst most values of recall, currently unknown to us.

5.12 AP comparison as function of size and difficulty between trained image pyramid and inference pyramid. Training on the image pyramid architecture does not always increase the performance. Only for the models $t310$ and $t710$.

5.13 Pr-curve for the models listed in table 5.4.

5.14 AP comparison as function of size and difficulty between image pyramids with selection criteria and without selection criteria. The selection criteria increases performance across all face sizes.

5.15 Pr-curve for the models listed in table 5.5.

A.1 The attributes in the wider dataset. All the models with the selection criteria are listed here including the baseline $t3$. We observe that our best models significantly perform better than our baseline on the attributed, normal blur typical expression, normal illumination, extreme illumination, no occlusion and partial occlusion. Additionally, our model under perform on the attributed typical and atypical pose and heavy occlusion.

A.2
Tables

5.1 Models with their specification and their abbreviation. The models with lower resolution perform better for the easy/medium subset while the models with higher resolutions perform better for the hard subset and overall AP. 31

5.2 Model with their specification and their abbreviation, these abbreviations will be used in the graphs in this section. Training increases performance for all resolutions. 32

5.3 Model with their specification and their abbreviation, these abbreviations will be used in the graphs in this section. Networks with large difference in resolutions perform better compared to networks with small difference in resolutions. Three branch networks perform equally or worse than their two branch counterparts. Networks with a 500 × 500 branch all have the same AP values. 34

5.4 Model with their specification and their abbreviation, these abbreviations will be used in the graphs in this section. Training on the image pyramid architecture does not always increase the performance. Only for the models 1310 and 1710. 36

5.5 Model with their specification and their abbreviation, these abbreviations will be used in the graphs in this section. The selection criteria always increases performance for all levels of difficulties. 1710-hc was expected to be higher than 17-e710-hc, which might be a cause of lower batch size. The difference between the model is bigger than 1% after 2 decimals. 38