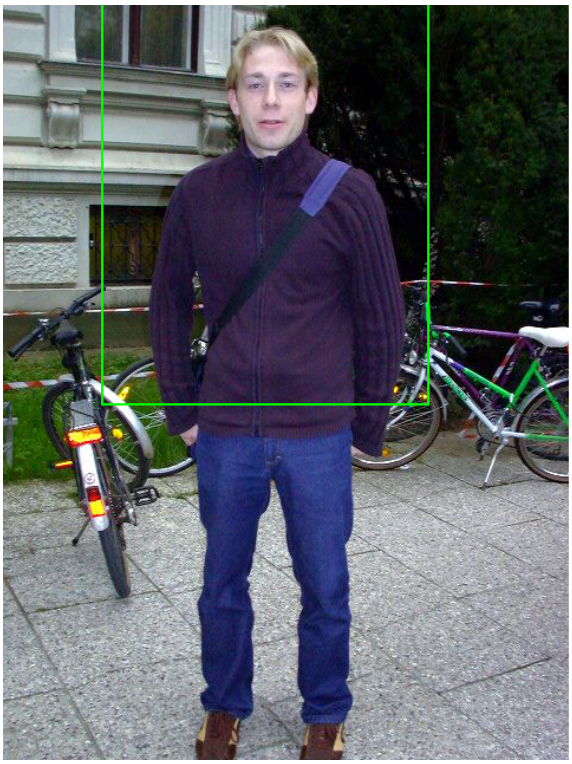
Upper Body Pedestrian Detection



VGIS9 - VGIS10 long Thesis
Group 15gr943 - 16gr1043
Institute for Electronics & IT
Aalborg Universitet
The 2nd of June 2016

AALBORG UNIVERSITET

Department of Electronic Systems
Frederik Bajersvej 7
9220 Aalborg Øst
Phone: 96 35 97 31
Fax: 98 13 63 93
http://www.sict.aau.dk/

Synopsis:

Traffic accidents involving pedestrians are a very serious matter as it has one of the highest risks to end with fatalities. Therefore, a program is needed to help detect pedestrians in a traffic scenario. In addition the system will be designed to be able to handle the most common occlusion problem, which is the lower part of the body and is therefore designed as an upper body detector.
The proposed upper body detector in this project is a CNN based system with three CNNs in a cascade. This is done in order to not only have a good performance, but also a system which is computationally lighter than a single deep CNN for possible real life scenario use. The cascade consist firstly of a very shallow CNN followed by a not as shallow CNN and lastly a deeper CNN for final predictions.
After the conducted experiments, it is shown that the performance of this upper body detector does not perform as well as a full-sized pedestrian detector. This was expected as the upper body is a subset of the full-sized pedestrian, but the results are bad enough to not be usable in a real life scenario. For that to happen further research on this topic is needed for improving on the feature extraction of upper bodies.

Title:
Pedestrian detection

Theme:
Computer vision

Project period:
Long thesis
VGIS9 - VGIS10
fall 2015 - spring 2016

Project group:
15gr943 - 16gr1043

Author:
Simon Mark Thomsen

Supervisor:
Kamal Nasrollahi

Number of copies: 1
Number of pages: 55
Number of appendix: 0
Finished: 02-06-2016

Simon Mark Thomsen

Preface

This report is written by group 15gr943/16gr1043 on 9th/10th semester at the department of electronic systems at Aalborg University in cooperation with Professor Robert Laganière at University of Ottawa, Canada. A thank you will be directed to Professor Laganière for hosting a stay at his laboratory (VIVA Lab) during this project.

Prior knowledge of basic machine learning is required to reading this report.

Reading Guide

There will through this report appear references to other peoples work, which will be summarised in a list at the end of the report. These references will be presented with the last name of the author(s) and the year of the work as [Last name, year]. The references are placed to give the reader a better sense of what the given reference covers. If the reference is placed in the text the reference covers the specific statement [In, 2016]. If it is placed at the end of a paragraph it covers the whole paragraph, or if it is placed after a paragraph it covers multiple paragraphs to a full section. [End, 2016]
[After, 2016]

Figures and tables are numbered according to which chapter they are in, e.g. the first figure in chapter 4 will have the number 4.1 and the second will be 4.2 and so on. The descriptive text for figures and tables will be found underneath the given figure or table.

Abbreviations

CNN	Convolutional Neural Network
Conv	Convolutional
ETH	Eidgenössische Technische Hochschule (Federal Institute of Technology)
FC	Fully Connected
FP	False Positive
FPS	Frames Per Second
FPPI	False Positives Per Image
HOG	Histogram of Orientated Gradients
LBP	Local Binary Patterns
LDCF	locally decorrelated channel features
MR	Miss Rate
NN	Neural Network
Pool	Pooling
ReLU	Rectified Linear Unit
SVM	Support Vector Machine
TP	True Positive
VGA	Video Graphics Array

Computer Set-up

OS	Ubuntu 14.04 LTS
Memory	23.5 GB
Processor	Intel Core i7-47-90 CPU @ 3.60GHz x 8
Graphics	GeForce GTX TITAN X/PCIe/SSE2
OS type	64-bit

Contents

1 Introduction **1**
- 1.1 Problem . . . 1
- 1.2 Objectives . . . 2

2 Previous Works **3**

3 Theory **11**
- 3.1 Convolutional Neural Network . . . 11
- 3.2 Training CNNs . . . 13

4 Design **17**
- 4.1 Baseline . . . 17
- 4.2 Upper Body Adaptation . . . 18
- 4.3 Proposed Network . . . 19

5 Implementation **21**
- 5.1 Data . . . 21
- 5.2 The First Network Proposal . . . 23

6 Results **29**
- 6.1 12-Network Results . . . 29
- 6.2 16-Network Results . . . 34
- 6.3 First Network Comparison . . . 39
- 6.4 Comparing Against State-of-the-Art . . . 39
- 6.5 State-of-the-Art Discussion . . . 39

7 Evaluation **43**
- 7.1 Conclusion . . . 43
- 7.2 Future Work . . . 44

Bibliography **45**

Introduction 1

According to Danmarks Statistik [2016], in the last 24 years the number of private cars owned has increased to 1.5 times the amount, as shown in Figure 1.1. This translates to an increase of more than 32.000 more cars each year. As well as this increase in the number of cars, the amount of traffic at busy highways and bridges has also increased 1.5 times, in just 20 years [Danmarks Statistik, 2014b]. This leads to the conclusion that Danes show a tendency of driving their cars more than previously. With this increase in car usage comes an increased risk of accidents.

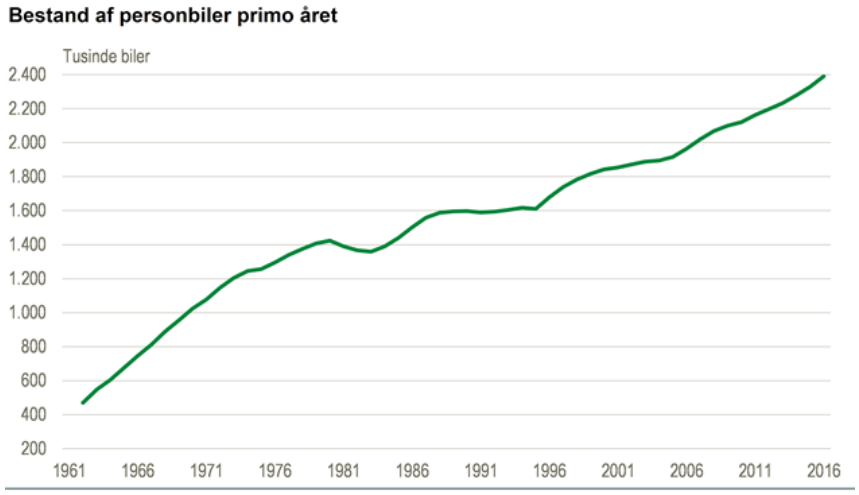


Figure 1.1. The number of owned cars in Denmark per annum in thousands [Danmarks Statistik, 2016]

1.1 Problem

Even though the number of traffic accidents in Denmark, has the previous years, mainly been falling in number there is still a substantial number of accidents which end in injuries or even death each year (See Figure 1.2 on the next page). Especially pedestrians are at high risk when involved in a traffic accident. Along with vans and motorcycles, pedestrians have one of the highest risks of a traffic related accident leading to death. Approximately one in 12 accidents involving pedestrians is fatal for the pedestrian. [Danmarks Statistik, 2014a]

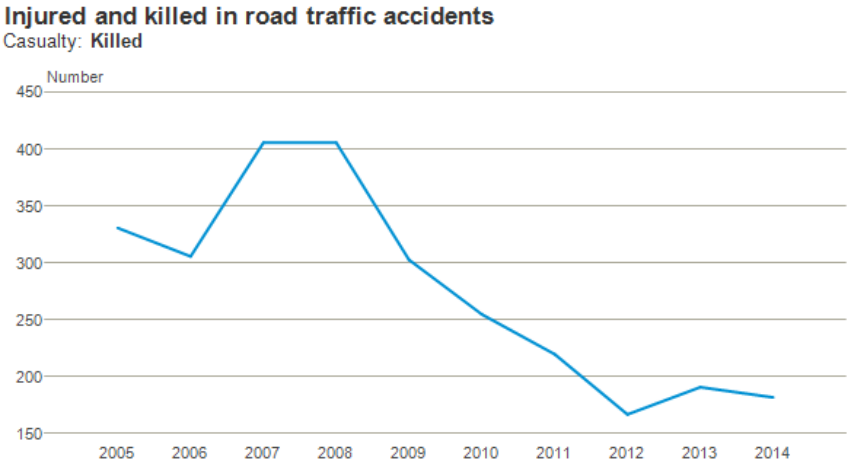


Figure 1.2. Number of people killed in road traffic accidents in Denmark per annum [Danmarks Statistik, 2014a]

A way to reduce the number of traffic related accidents, is to design a system for cars which assists the driver by increasing awareness of surrounding pedestrian, bicycles, mopeds, and motorcycles, whilst driving.

There has been a lot of research in the pedestrian detection field of computer vision, but in order to perform well in real life scenarios, the detector has to be robust against all kinds of poses, occlusion, and rotation.

1.2 Objectives

As of the discussion in the previous section, it has been chosen in this report to present a pedestrian detector specialised in detecting only the upper body, instead of doing full-sized pedestrian detection as previous works (explained in the next chapter on page 3). This is done to overcome any segmentation there might be of the legs of any pedestrians. The reason behind choosing to only focus on segmentation of the lower body of pedestrians, is that statistics from the Caltech pedestrian dataset shows, approximately 70% of all occlusions of pedestrians is on the lower part of the body Dollár et al. [2012].

1.2.1 Delimitation

The work in this project will be focused on designing a pedestrian detector (i.e. upright standing or walking people) and not a person detector.

1.2.2 Problem Statement

The following problem statement has been set and will be the foundation for the design of the following work.

How is it possible to design a system for pedestrian detection,
which achieves good performance,
as well as handles segmentation of pedestrians' lower body?

Previous Works 2

In this chapter, work previously done by other researchers will be described to give the reader an idea of the research this project builds on. This includes recent works on different detectors, like pedestrian and face detectors.

There are many ways to do pedestrian detection. Both using classical features, convolution and neural networks. The work by Yang et al. [2015b] uses classical features, and shows a way to use prior knowledge of pedestrians in order to improve performance. Traditionally, Haar-like features are used as a specific filter for a specific channel, and does not take into account any cross channel features or symmetric aspects of pedestrians, there might be. This work proposes a way to combine Haar features with symmetry and cross-channel features, to improve performance.

Symmetric features are computed by calculating the difference between the two rectangular filter responses positioned symmetrically around an axis in a subregion of the pedestrian (See the subregion in Figure 2.1 (c)). The cross-channel features are calculated the same way, however, instead of the rectangles being positioned symmetrically around an axis they have the same position and size, but in a different channel. In Figure 2.1 (b) it is possible to see the top symmetrical (top) and cross-channel (bottom) features. In this work feature extraction is done by histogram of orientated gradients (HOG), LUV, local binary patterns (LBP), and locally decorrelated channel features (LDCF), followed by the described feature pooling and classified by a decision forest.

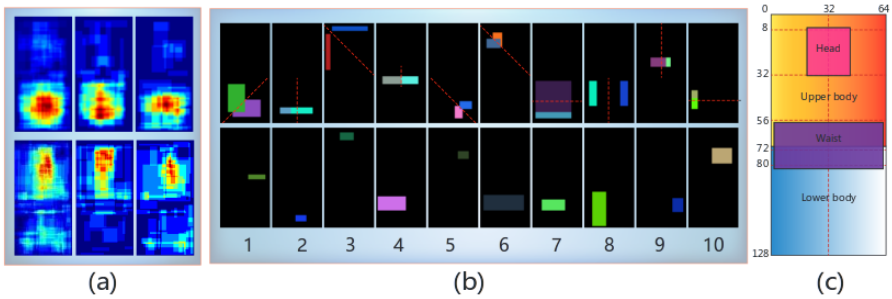


Figure 2.1. (a) Heat maps of the top 100 of all, symmetrical, and cross-channel features from left to right respectively (b) Top 10 symmetrical (top), and cross-channel (bottom) features (c) The subregions of a pedestrian [Yang et al., 2015b]

Another way to use HOG with supplementary features, is to include depth, as done by González et al. [2015]. They incorporate depth by using LIDAR information to construct a dense depth map from a point cloud, and use this map along with HOG and LBP for feature extraction. Classification is done by splitting pedestrians into 4 orientations (90 degrees each) in order to have separate models for the pedestrian's front, left, right, and back side. The orientations are then further spilt into smaller patches in order to have part specific models for occlusion handling, which then is classified by using a decision forest. Each note in a tree represents one of the smaller patches and is individually classified by a support vector machine (SVM) trained for the specific patch. Their full pipeline can be seen in Figure 2.2.

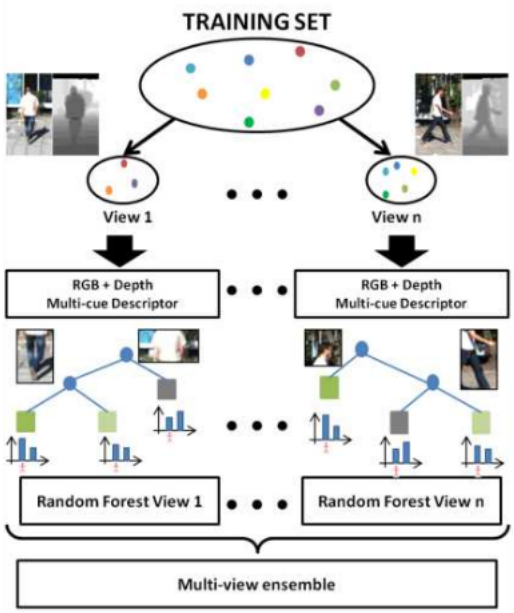


Figure 2.2. The pipeline proposed by González et al. [2015] using RGB and LIDAR information

Zhang et al. [2015] also looks at pedestrian using HOG and Haar-like features, but instead of coming up with supplementary features, they explore what impact the choice of feature bank has on performance. They present a pedestrian detector using solely HOG and LUV for low-level features extraction. From these feature maps, feature pooling is done by sum-pooling with a set of rectangular regions to construct feature vectors. This set of rectangular regions is taken from other previous works and filtered into a set of the best performing filters. For classification they use a decision forest, trained with Adaboost, and the full pipeline of their system can be seen in Figure 2.3.

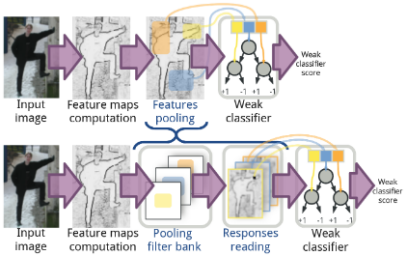


Figure 2.3. The pipeline of Filtered Channel Features pedestrian detector [Zhang et al., 2015]

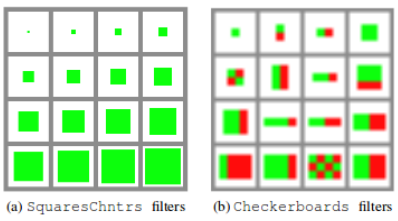


Figure 2.4. Examples of filters from Filtered Channel Features [Zhang et al., 2015]

Their work successfully links the previously used independent filter banks together in a unified system by doing intensive testing of combination of these filter banks, as well as testing different sizes of filter banks to find the optimal one. See Figure 2.4 for example of said filter banks.

Benenson et al. [2012] take a different approach to contribute to pedestrian detection. Rather than trying to improve performance, they instead they look at ways to increase the speed of state-of-the-art detectors using HOG and LUV features. During their tests, they find that roughly half the computation time is spent on image resizing and generation of integral images, and that the other half is spent on computing feature responses and classification scores. To circumvent this wasted time on resizing images, they present a way to use different sized models (one for each scale). However, this dramatically increases training time, because each model needs to be trained individually for its specific scale. To solve this they propose a way to, instead of training N models, train $\frac{N}{K}$, and then at test time transform these trained models into approximations of models, for all the needed scales. To see the comparison between the different approaches see Figure 2.5.

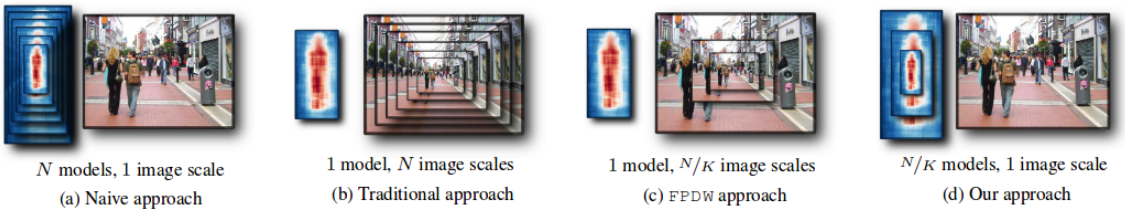


Figure 2.5. A comparison between the different approaches discussed i Benenson et al. [2012]

Furthermore, they use an approach called stixel world modelling, which uses depth information to be able to approximate "sticks above the ground". This allows them to reduce the search space for their model to this smaller region of the image, and thereby gain an increase in speed. This stixel world modelling can be seen on Figure 2.6, as well as detection examples.

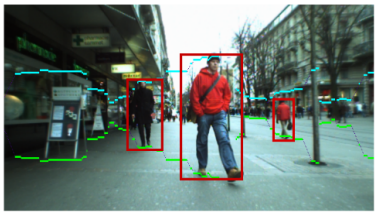


Figure 2.6. The stixel world modelling can be seen in the image as the blue and green lines encapsulating the region of interest, and detections as red boxes [Benenson et al., 2012]

Instead of using HOG for low-level feature extraction, it is also possible to use convolutional filters, which is done by Yang et al. [2015a]. In their work they take the first convolutional layers from a pre-trained convolutional neural network (CNN) and uses these for feature extraction. Then they use a decision forest by taking the single pixel values outputted from the convolutional filters as features for the trees. The convolutional layer model they use can be seen in Figure 2.7 on the next page.

Another way of designing a pedestrian detector is to use CNNs for both feature extraction and classification. CNNs come in many shapes and sizes like the proposal from Angelova et al. [2015], which contains multiple architectures. They have proposed a pedestrian detector as a cascade classifier based on AlexNet [Krizhevsky et al., 2012]. Because of

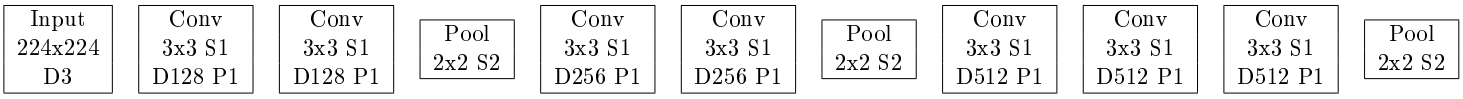


Figure 2.7. The convolutional layers used for feature extraction done by Yang et al. [2015a]

AlexNet's big size and computational complexity, this network is very slow. To make up for this, this work used it at the end of a cascade to speed up the detection speed of the otherwise slow deep CNN to real-time, as a lot of patches get rejected in the early stages.

Their network is a three staged cascade, consisting of three networks, from which, the two last are CNNs. The first network is very fast Adaboost approach, using HOG, LUV, and decision trees. This first network is originally created as a full pedestrian detector, but in order to keep a good recall after first network, Angelova et al. [2015] reduced the amount of stages in the cascade to only 10% of the original number.

After this fast proposal network, they feed the output into the first medium seized CNN (The second network in the cascade), which can be seen on the top of Figure 2.8, to further reduce the amount of windows. This is followed by the AlexNet inspired last CNN, which is a deep network for the final classification and can be seen on the bottom of Figure 2.8.

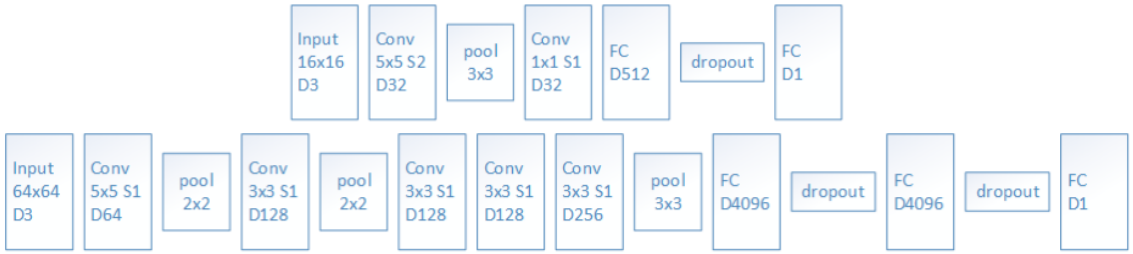


Figure 2.8. The second (top) and third (bottom) network in the Angelova et al. [2015] cascade

Instead of striding towards real-time application, Tian et al. [2015] work to improve performance by including semantics task in their CNN. Their proposal to improve pedestrian detection is to handle the problem of using a single binary classifier for all pedestrians, and instead jointly optimise pedestrian detection with semantic tasks. This is done by assigning attributes to both the pedestrians and the scene. Examples of these attributes are 'backpack', 'gender', and 'dark-trousers' for pedestrians and 'vehicle' and 'tree' for the background (See Figure 2.9 on the next page for (a) data generation and (b) network structure). This increases the network's ability to handle these complex pedestrian variations, because it does not need features that cover the full variance of pedestrians in one.

Like this approach, Li et al. [2015] used a similar approach, though instead of splitting pedestrians by attributes, their work presents a way to handle features of pedestrians being dramatically changed from high resolution to low resolution. To differentiate between low and high resolution pedestrians, a small-sized and a large-sized sub-network is integrated into the network, as seen in Figure 2.10 on the facing page.

Their network consists of shared convolutional layers for early feature extraction. After

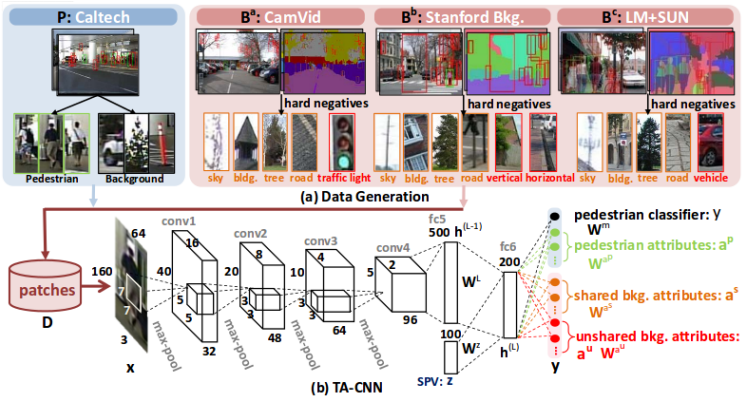


Figure 2.9. The pipeline from Tian et al. [2015] showing (a) the data generation for for training the detector and (b) the full structure for their network

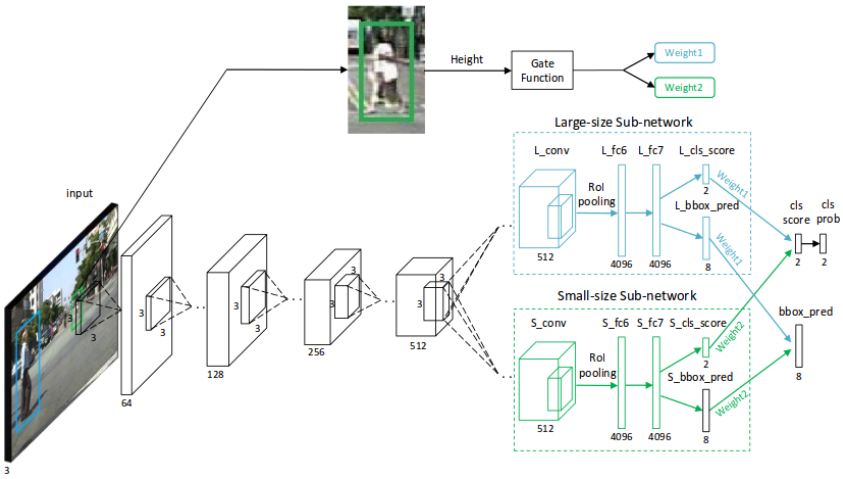


Figure 2.10. The pipeline from Li et al. [2015] showing the network and the gate function to adjust the weights for each sub-network

that, the output is fed into two sub-networks, which each computes class scores and bounding box prediction. The class scores and bounding box predictions are then weighted according to a gate function, looking at the height of the candidate window to output a final prediction.

A different approach to pedestrian detection is used by Ouyang and Wang [2013]. Here a joint deep learning approach is taken. This is done by making a network which jointly learns feature extraction, part deformation handling, occlusion handling, and classification. Their network can be seen in Figure 2.11 on the next page.

Part detection is done after a global feature extraction, which is a standard convolutional layer. But since different parts of pedestrians have different sizes, variable sized filters are used for each part. The parts are put into three levels with level one being for small parts, level two for medium parts, and level three for large parts. Each part in level two and three consist of a combination of parts from a lower level. The total number of parts ends up at 20 and can be seen in Figure 2.12 on the following page.

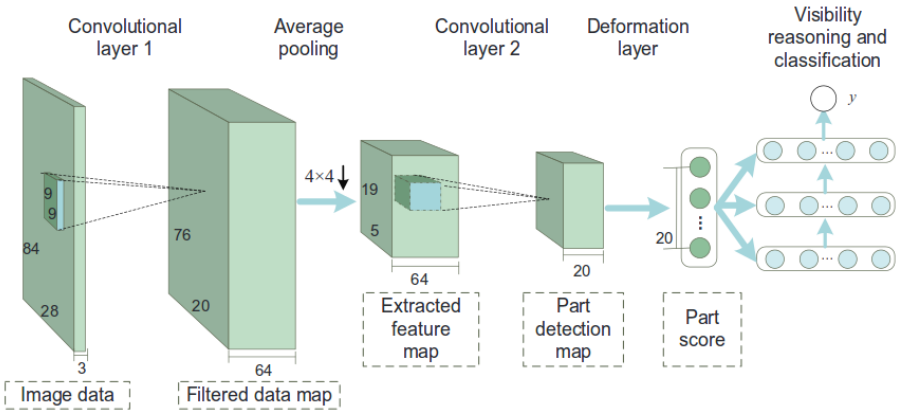


Figure 2.11. The pipeline proposed in Ouyang and Wang [2013] showing the different layers

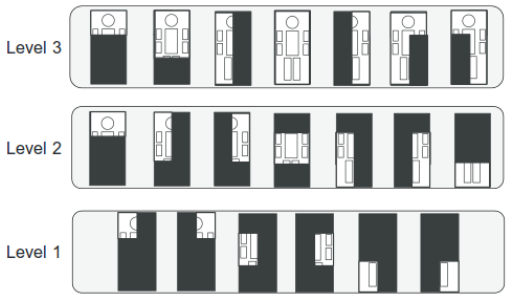


Figure 2.12. The parts used in Ouyang and Wang [2013] at its respective level

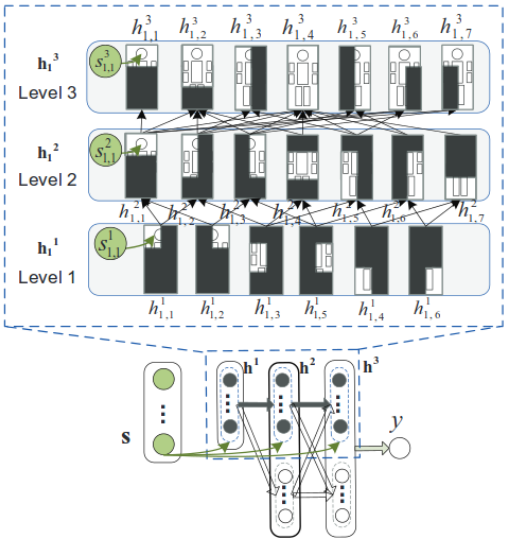


Figure 2.13. The model for visibility reasoning and detection label estimation [Ouyang and Wang, 2013]

A deformation layer is then added to individually calculate part scores from the prior part detection maps by comparing it with learn deformation maps for each part. The part scores are then given to a visibility reasoning layer, which forwards scores of lower level parts to the higher level parts, of which they are a part of, to output a final class score as seen on Figure 2.13.

For the purpose of looking into architectures, which are not pedestrian detectors to attain knowledge of other fast CNN structures the work from Lit et al. [2015], which resembles Angelova et al. [2015] cascade structure, is looked at. They have proposed a cascade of CNNs for face detection. This is again done in order to have a high performance by using CNNs and still keep the network computationally light, by using them in a cascade by quickly rejecting background regions, which are easily classified as background by shallow classifiers. The more challenging background patches will then be evaluated in the later and deeper CNNs at the end of the cascade. The cascade's classification networks can be

seen in Figure 2.14.

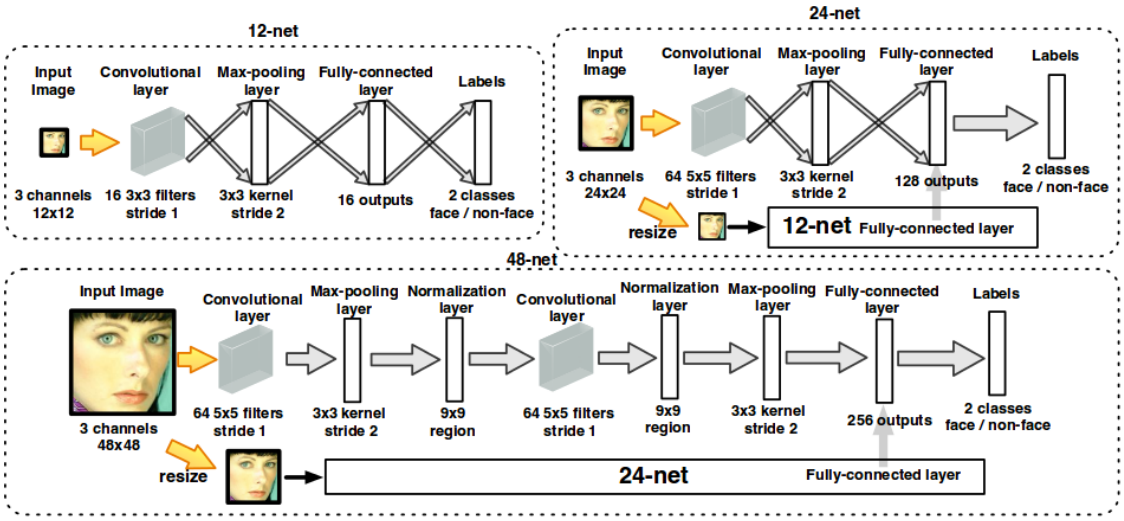


Figure 2.14. The classification networks proposed in Lit et al. [2015]

In addition to the classification CNNs the paper also proposes to use CNN for calibration of the detection windows, to get a more precise detection, which can be seen in Figure 2.15. This is done by training the CNN with 45 classes, which each represent a scaling and/or a translation of the ground truth detection window, and thereby be able to predict the correct position of a detection window.

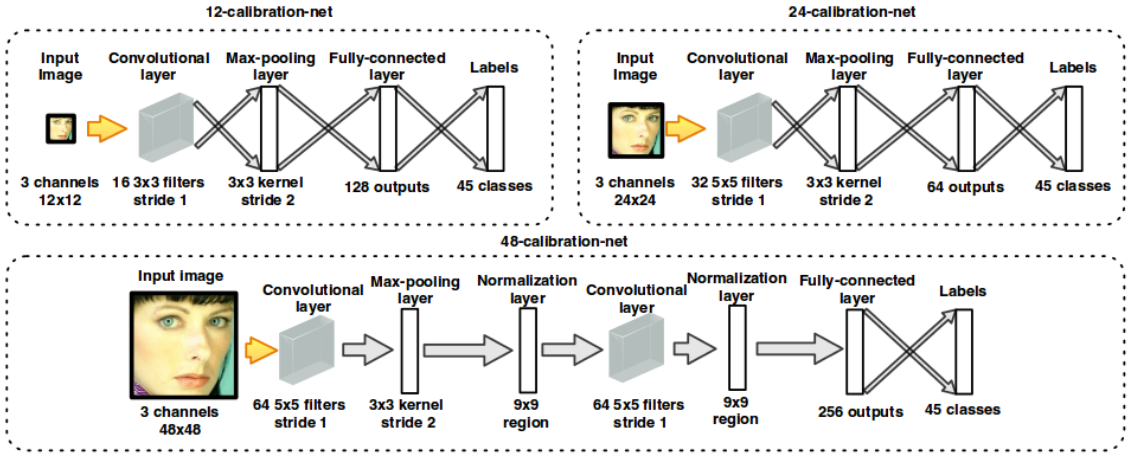


Figure 2.15. The calibrations networks proposed by Lit et al. [2015]

As seen in Figure 2.14 and 2.15, this paper's cascade consist of three CNNs for classification and three CNNs for calibration. The structure of the cascade is that a classification network is followed by a calibration network so that the input to the next deeper classification network is more precisely placed detections, and can be seen in Figure 2.16 on the next page.

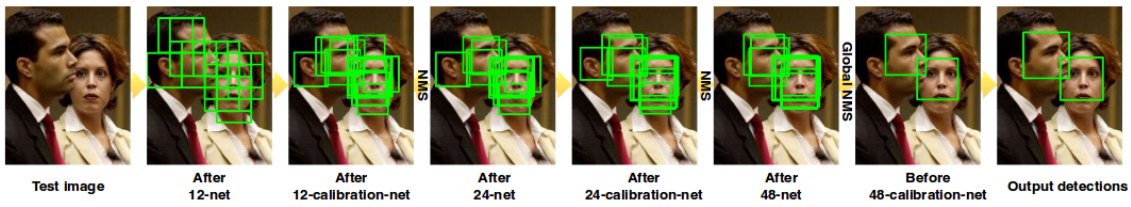


Figure 2.16. The full pipeline from Lit et al. [2015]

Theory 3

In this chapter CNNs and ways to optimise a CNN during training, which improves speed and performance will be described to give an idea of the contribution of these schemes.

3.1 Convolutional Neural Network

CNNs is a specialized neural network (NN). The reason not to use a regular NN is because NN does not scale well. E.g. take an image with a size of 3x32x32 (3 colour channels, 32 pixels wide, 32 pixels high) A neuron in a NN would have $3 \cdot 32 \cdot 32 = 3072$ weights. For an image with the size 3x256x256 a neuron would have $3 \cdot 256 \cdot 256 = 196,608$ weights. This number then needs to be multiplied with the number of neurons, which will lead to overfitting.

CNNs take advantage of the fact that the input consists of an image and by knowing that the neurons are arranged in three dimensions (width, height and depth) and by making the neurons in each layer connect only to a small region of the layer before it. Though this is not enough to reduce the amount of parameters to a reasonable level. One way to dramatically reduce the number of parameters is by making the assumption, that if one patch's feature is useful to compute at the spatial position (x_1,y_1), then it should also be useful to compute in a different position (x_2,y_2). This means that all the neurons that share the same depth use the same filter and this greatly reduces the number of parameters.

3.1.1 Layer Types

Traditional CNNs consist of a handful of different layers which each have a specific role within the network. Each of these layers will be described in the following section.

Input Layer

The input layer is the first layer usually containing an image with raw pixel values, e.g. a greyscale or RGB image.

Convolutional Layer

Following the input layer are a number of convolutional layers which computes the output neurons from their receptive field[1] in the previous layer, with a set of learnable filters, that will activate when they see specific types of features. These filters are stacked along the depth dimension to form the output volume.

The convolutional layers have three hyperparameters, which control the size of the output volume: depth, stride, and zero-padding.

Depth is the number of neurons in the convolutional layer that connects to the same region of the input volume.

Stride is how much each filter is moved in the previous layer between each neuron in the current layer. Low strides leads to heavily overlapping receptive fields.

Zero-padding is to pad the input volume with zeros on the border to preserve the spatial size of the input.

The fit of a convolutional layer can be calculated by (3.1)

$$fit = \frac{W - F + 2P}{S} + 1 \tag{3.1}$$

W	Input volume size
F	Receptive field size
P	Zero-padding
S	Strides

If the fit is not an integer, then the kernel does not "fit" across the input volume in a symmetric way. This means that some pixels might be skipped because that are positioned outside of what "the fit allows".

Example:

- Image size: 227x227x3
- Receptive field size: 11
- Strides: 4
- Zero-padding: 0
- Convolutional layer depth: 96

$$\frac{227 - 11 + 2 \cdot 0}{4} + 1 = 55 \tag{3.2}$$

As seen from (3.2) the fit of the convolutional layer will be 55 output neurons.

Activation Function Layer

The activation function layers purpose is to apply an elementwise activation function on the output from a convolutional layer. Usually activation functions are a type of non-linear function which could be $\tanh(x)$ or $\max(0, x)$.

[1] The receptive field of a neuron is the local region to which it is connected

Pooling Layer

A pooling layer is placed after convolutional layers, which performs a downsampling operation along the spatial dimensions (width, height), and is typically implemented as max-pooling, but could be average-pooling or other pooling schemes. The reason for using a pooling layer, is to reduce the number of parameters, and the complexity by reducing the amount of neurons. It is common to use a pooling layer with a kernel size of 2x2 and with a stride of 2.

Fully Connected Layer

After all the convolutional and pooling, the layers fully connected layers are located. Their purpose is to compute the class scores for the input image and work as a traditional NN.

[Karpathy]

3.2 Training CNNs

Training CNNs is done the same way as a traditional NN. To do this, a loss function will be needed, which could be defined by the negative log likelihood criterion as seen in (3.3). [Karpathy]

$$L_i = \sum_j y_{ij} \log(\sigma(f_j)) + (1 - y_{ij}) \log(1 - \sigma(f_j)) \tag{3.3}$$

L_i	Loss for the ith data
j	Class index
y_{ij}	Labels for the data (1 for positive, 0 for negative)
f_j	Score vector

Minimised this loss function is desired to make the network as good as possible. This can be done by using stochastic gradient descent which for a N parameter problem is defined as in (3.4). [Ruder, 2016]

$$\theta_n = \theta_n - \alpha \frac{d}{d\theta_n} L(\theta_0, \ldots, \theta_N) \tag{3.4}$$

θ_j	The jth parameter for optimisation
α	Learning rate
$L(\theta_0, \theta_1)$	Cost function dependent on θ_0 and θ_1

Furthermore there are many ways to improve upon the training some of which will be described in the following sections.

3.2.1 Dropout

Dropout [Srivastava et al., 2014] is a training scheme aimed to prevent networks from overfitting. It is a common problem that large CNNs overfit because of lack of variance in

training data. This makes the network adapt too much to the training data which leads to a fall in testing accuracy because the network becomes too specialised to the specific training data.

Because large CNNs are slow it is not ideal to use multiple different networks and average their results in the end, which is why dropout is presented. Dropout works by randomly dropping units (Ignoring them as by setting their output value to zero as seen in Figure 3.1 and 3.2) in the fully connected layers of networks. This makes the network seem "thinner" and because of the dropping of random units and creates a high number of these "thinner" networks with units combined multiple ways. It is then simulating having a lot of small networks which during testing time is combined to one big, unthinned, network (i.e. model averaging).

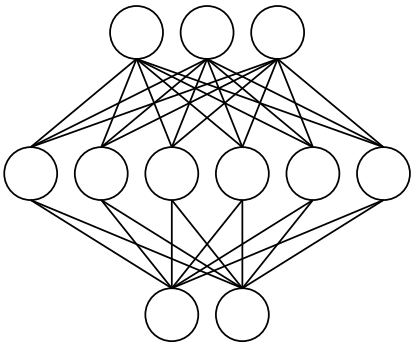


Figure 3.1. Fully connected layer without dropout

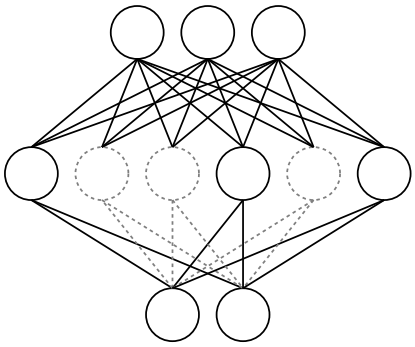


Figure 3.2. Fully connected layer with 50% dropout

3.2.2 Random Dropout

Random dropout [Fukui et al., 2015] is an addition to dropout which works in the same way as dropout but instead of having a fixed dropout percentage (usually 50%) the dropout percentage changes from epoch to epoch. For example it could be set to vary between 30 and 70% which can be seen in Figure 3.3 and 3.4.

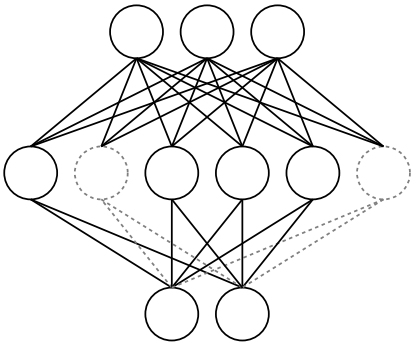


Figure 3.3. Fully connected layer without random dropout, 30 to 70%, at epoch n (33% dropout)

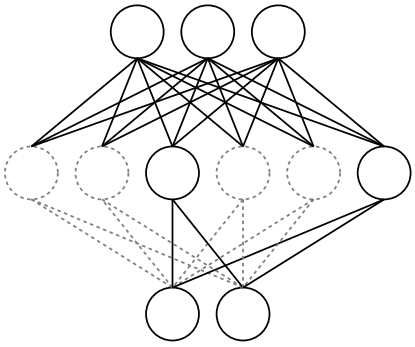


Figure 3.4. Fully connected layer without random dropout, 30 to 70%, at epoch $n+1$ (66% dropout)

3.2.3 Rectified Linear Unit Layer

Rectified linear unit layer (ReLU-layer) does according to Krizhevsky et al. [2012] improve the training speed of CNNs by several times compared to hyperbolic tangent. ReLU-layers are calculated like (3.5) and is illustrated in Figure 3.5.

$$f(x) = \max(0, x) \tag{3.5}$$

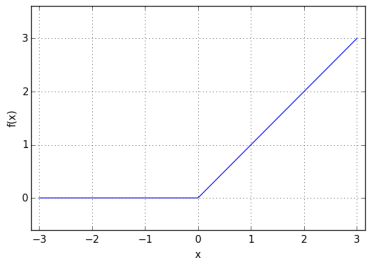


Figure 3.5. ReLU activation function

3.2.4 Positive and Negative Data Ratio

Because it is common to have many times more negative than positive data when training, the training batches can become very skewed in terms of the ratio between positive and negative data. For example if batches containing 128 patches are used and the ratio between positive and negative data is 1:100 respectively, each batch will only contain about two positive images with a uniform sampling. This will drastically reduce the detectors ability to distinguish positives from negatives. It is therefore advantageous to have a fixed ratio between positives and negatives of at least one positive for every three negatives. [Farfade et al., 2015]

Design 4

This chapter will firstly contain a description of the chosen baseline detector followed by a discussion of how to adapt this detector to upper bodies. Lastly there will be a description of the design of this project's proposed network for upper body detection, based on the state-of-the-art in full sized body pedestrian detection and other works with fast deep networks.

4.1 Baseline

Since this upper body detector should be used instead of a full sized pedestrian detector, Angelova et al. [2015], as described in section 2 on page 5, is the main reference network and is reimplemented here to be used for performance comparing.

4.1.1 Cascade and Network Structures

As stated previously Angelova et al. [2015] consists of a cascade with three networks. The first network in the cascade is a reduced version of the network by Benenson et al. [2012] described in section 2 on page 4. Usually the network contains a 2000 stage decision forest but as stated in the original paper this number is reduced to 200 to keep a good recall at this point in the cascade.

The second network in the cascade is a small CNN which can be seen in Figure 4.1. The third and last network in the network is a deep CNN which has a Alexnet [Krizhevsky et al., 2012] like structure, and can be seen in Figure 4.2.

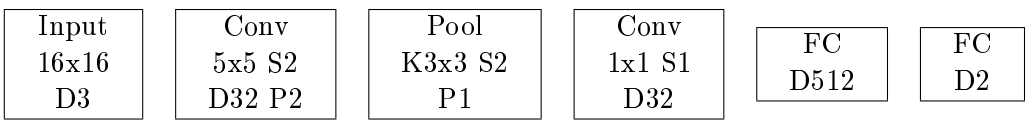


Figure 4.1. The second network in the Angelova et al. [2015] cascade

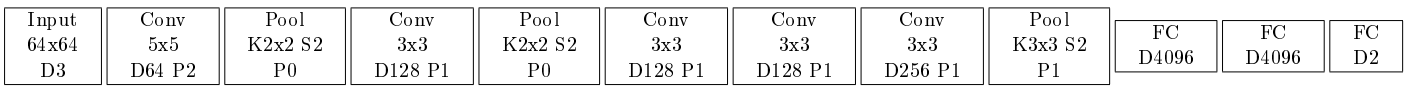


Figure 4.2. The third and last network in the Angelova et al. [2015] cascade

4.1.2 Results

The results of the reimplemented baseline detector can be seen in Figure 4.3. It is seen that the performance of this implementation does not perform up to standards with the original works. This is likely due to improper training and not using the pre-trained networks which is mentioned in the original works to improve performance.

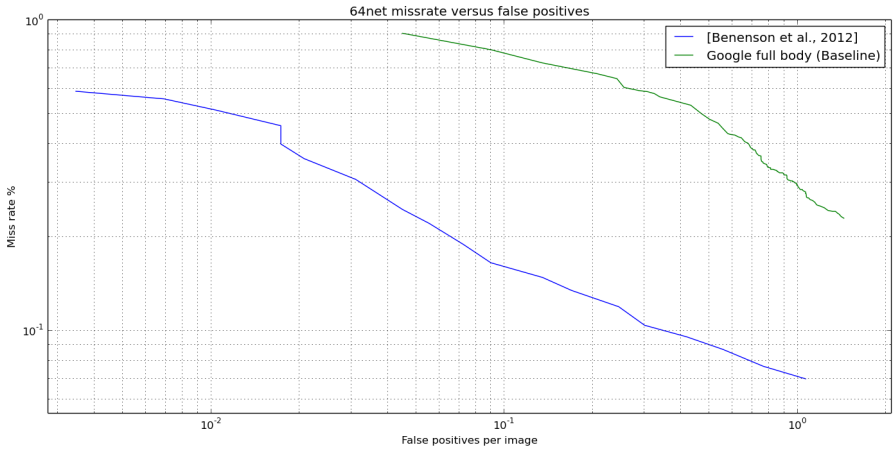


Figure 4.3. Performance of the baseline detector for full sized pedestrians tested on INRIA dataset

In order to really see the how upper bodies' performance is compared to full sized pedestrians detection a baseline detector should reimplemented with state of the art performance. Since this is not the case, the results of this work should only be taken as comparison between the curves which is assumed to have the same relationship at a better performance level.

4.2 Upper Body Adaptation

As stated above it is chosen to base the proposed upper body detector on the work in Angelova et al. [2015] and in addition to that, base it on the network proposed by Lit et al. [2015] (See section 2 on page 8), a pedestrian detector and a face detector, respectively, both with CNN cascades. The reasons behind this is that both of these network runs in real-time with state-of-the-art performance.

In order to adapt Angelova et al. [2015] to find upper bodies, some of the papers' principles must be changed in order to make it fit the new problem. One of the problems is that the detection boxes does no longer have the same ratio. State-of-the-art pedestrian detectors use aspect ratios anywhere from 0.34 to 0.5 [Dollár et al., 2012], a detector for upper bodies has to be even higher, specifically in this project the aspect ratio for upper body detection is set to 0.8 i.e. four wide to five high.

4.2.1 Adaptation Changes

Since of the scope of this project, the first Adaboost network is not changed to detect upper bodies, but used as a full-sized pedestrian detector feeding the output to an upper body classifier for some of the test found in chapter 6 on page 29. Furthermore, a new

network is proposed, which is described in section 4.3. This network is a combination of two paper mentioned in section 4.2 on the facing page.

The second network for the upper body adapted cascade can be seen in Figure 4.4 which includes changes in the input layer's size and the pooling layer's padding to make up for the first change. For the third network the only change is the input layer's size and can be seen in Figure 4.5.

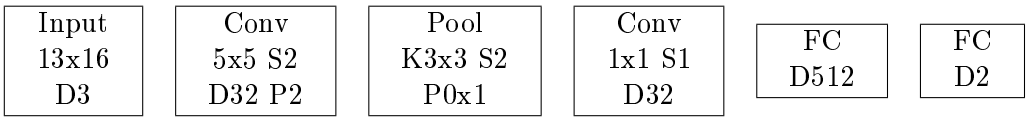


Figure 4.4. The second network in the upper body adapted cascade

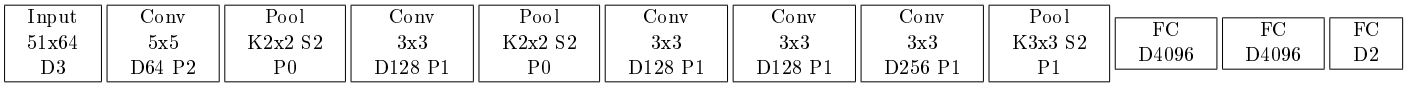


Figure 4.5. The third and last network in the upper body adapted cascade

4.3 Proposed Network

In the following sections, some possible solutions for replacing the first network in Angelova et al. [2015] are presented based on the paper Lit et al. [2015] to give an idea of the thought process behind the made choices.

The first network in the cascade from Lit et al. [2015] is a tiny 12x12 network which can be seen in Figure 4.6. It is wanted to keep one of the sides at 12 pixels or 16 pixel as in the original implementations. This makes the probable resolutions able to be 10x12, 12x15, 13x16, or 16x20.

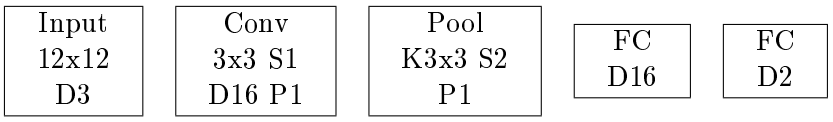
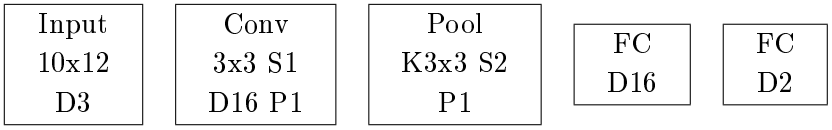


Figure 4.6. First network in the Lit et al. [2015] cascade

It is chosen to use 12 and 16 as the longest end which leaves 10x12 and 13x16. Both of these sizes will be tested to compare performance, before a choice between them, is made. However, by using the resolutions 10x12 and 13x16, several problems arise, such as the pulling layer, as you can not take the half of 13 pixels. Therefore, some of the proposed networks to replace the first in the cascade are presented in the next sections, to show some of the problems and to give a basic idea of how to solve them.

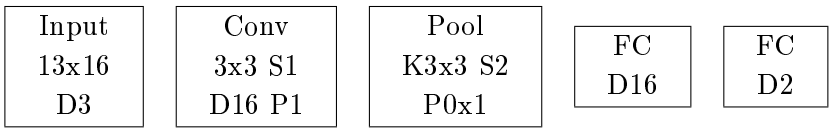
4.3.1 Version One

For input size 10x12 nothing needs to be changed as seen below.



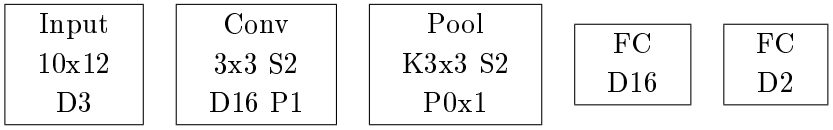
4.3.2 Version Two

For input size 13x16 the pooling layer's padding needed to be changed to be 0x1 and can be seen below.



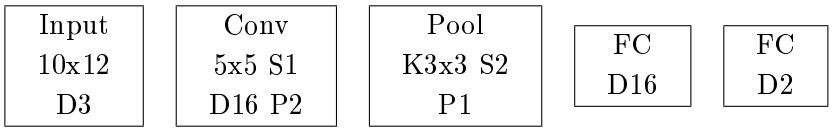
4.3.3 Version Three

As seen in the Angelova et al. [2015], the second network has a stride of two for the proposed convolutional layer and should be tested to see if it increase performance. This also makes the padding for the pooling layer to be 0x1 as seen below.



4.3.4 Version Four

The 3x3 kernel in the convolutional layer can be replaced by a 5x5 kernel with 2x2 padding, which can be seen below.



4.3.5 Depth

The depth of the convolutional and fully connected layers are in all of the above proposals set to 16, as in the original paper, but could be changed to an arbitrary number, and are not necessarily the same, as it is seen in the tested network shown in the next chapter.

Implementation 5

From these few explained first networks, all of the tested networks for upper body adaptation will be shown in this chapter, but firstly, the data collection for training the networks, will be described.

5.1 Data

This section will contain information about all the data used in this project. Firstly, a section about the data collection, followed by a description of how the dataset's data is augmented to fit the current problem, and lastly how the data used.

5.1.1 Data Collection

Data for training and testing of the network will be extracted from multiple datasets. Namely a self created, INRIA, and ETH dataset properties of which will be described briefly.

Self-created Dataset

This is a pedestrian dataset, created by VIVA Lab, made up only of images from surveillance cameras inside food chain stores from a top view angle. In this dataset many of the annotated pedestrians have occluded legs which fits with the need of a upper body detector. The dataset contains 6,500 images with 2,300 bounding boxes. An example from this dataset can be seen in Figure 5.1 on the following page.

INRIA Dataset

The INRIA dataset is a pedestrian dataset containing still images with different resolutions taken at many different locations. Annotations are only put on upright people with a height of more than 100 pixels. The size of the training examples from this dataset is 1237 pedestrians and an image from the dataset can be seen in Figure 5.2 on the next page. [Dalal and Triggs, 2005]

ETH Dataset

ETH is a pedestrian dataset of stereo video, taken from a car, with a resolution of 640x480 and a frame rate of 13-14 FPS. The annotations for this dataset is put on pedestrians with

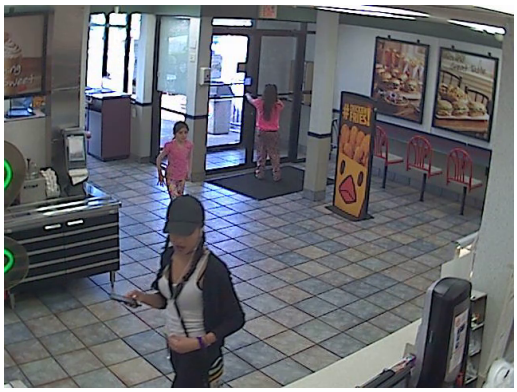


Figure 5.1. An example image from the self created dataset

Figure 5.2. An example image from the INRIA dataset [Dalal and Triggs, 2005]

a minimum height of about 48 pixel, but for testing bounding boxes with a height lower than 60 pixels have been removed. ETH have 4,500 with 17,800 bounding boxes and an example image from the dataset can be seen in Figure 5.3. [Ess et al., 2008]

Figure 5.3. An image taken from the ETH dataset[Ess et al., 2008]

5.1.2 Dataset Augmentation

All of these dataset contain full-sized pedestrians, except the self-created one. Because of this difference in the project's pedestrians detection and traditional pedestrian detection (i.e. upper body against full body), the datasets had to be modified to fit the problem.

The first measure for converting the data, is to fit the bounding boxes to only cover the upper body of the pedestrians. Due to the way the width of bounding boxes for walking pedestrian oscillates from frame to frame, it is chosen to use the height of pedestrians to find the upper body part, as it changes more gradually [Dollár et al., 2012]. Getting the upper body from the height of a pedestrian is done by cutting it in half and then changing the width to get the desired aspect ratio of 4:5.

5.1.3 Dataset Usage

The acquired upper bodies are then slit into multiple sets of data serving different purposes which will be presented in the following sections.

Training Set

The training data is cropped images of upper bodies and background to use for training the networks. The positive data consist of images from a subset of the positive training data from INRIA, a subset from the self-created data, and all the positive data from ETH. The same positive data is used for all the networks, and the negative data is randomly sampled patches from images of background, when used for training of the first network. For the second network, negative patches are mined from background images by taking hard examples (false positives) from the previous network. The images used for mining negative image are from INRIA training set and the Pascal dataset [Everingham et al.].

Validation Set

A set of data is used for testing during training to make it possible to follow the performance of the training on a separate set. This is done in order to know when the network performance starts converging, to keep an eye out for overfitting, and if so do early stopping. This set is also pre-cropped images, and is a small subset of the self-created dataset, ETH dataset and the negative patches.

Testing Set

The testing set is a set of full images used for testing performance such as precision, recall and false positives per image and optimisation of thresholds. This set is the all the positive images from INRIA testing.

This concludes the description of the data mining and handling thereof which will be used for training of all the networks proposed in the upcoming sections.

5.2 The First Network Proposal

The purpose of the first network, is to reduce the amount of detection windows for the later networks, but it is also important that it keeps a high recall rate so as not to reject too many positive detection windows at this stage.

Firstly 12-network structures, which are networks where the input image's height, is 12 pixels, will be described, and secondly the 16-network structures, which have a height of 16 pixels, will be described. The testing of these networks will all be found in chapter 6 on page 29.

In this chapter every time the Google network is mentioned it is a reimplemented version of it its second network in the cascade, but adjusted to detect upper bodies, described in section 2 on page 5 (Note that it is not the first network).

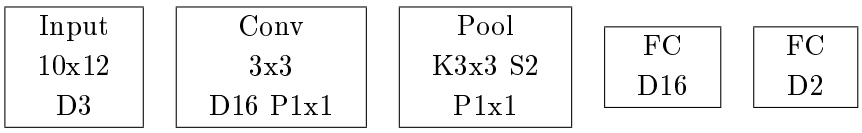
5.2.1 12-Network Structures

This is a list of all the tested structures with an input image size of 10 by 12 pixels (10 in width and 12 in height), except for the Google network implementation, which is shown here for comparison reasons between 12- and 16-networks. This is to give an understanding of the differences in the structures of the networks, when inspecting the following performance graphs and tables. In Table 5.1 is a list of abbreviations used in the following sections.

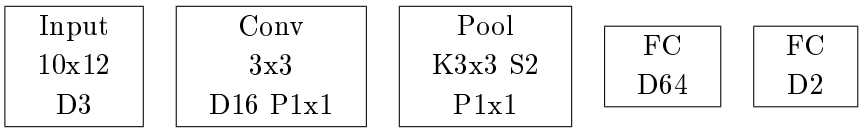
Abbreviation	Description
Conv	Convolutional layer
D	Layer depth
FC	Fully connected layer
Input	Input layer
K	Kernel size
P	Padding amount
Pool	Pooling layer
S	Stride size

Table 5.1. Abbreviation list for network structures

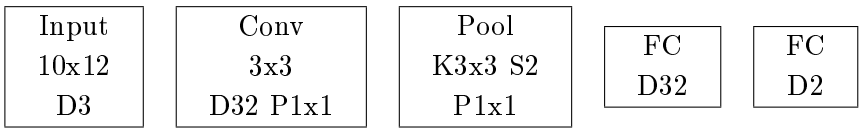
(1) 16x3x3 > FC16



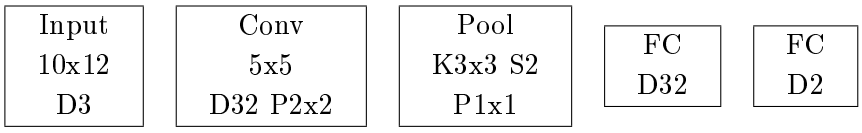
(2) 16x3x3 > FC64



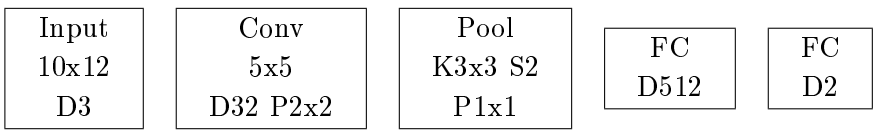
(3) 32x3x3 > FC32



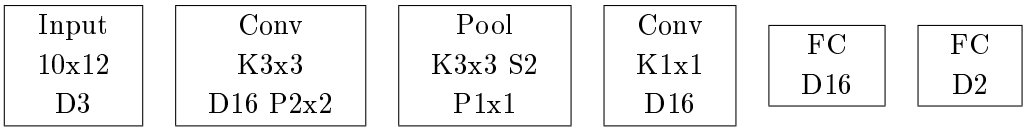
(4) 32x5x5 > FC32



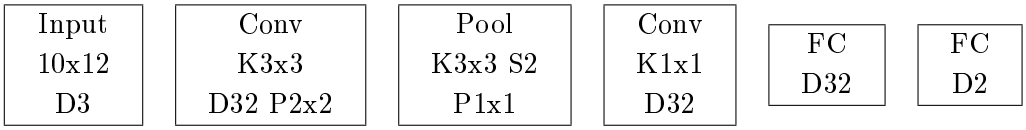
(5) 32x5x5 > FC512



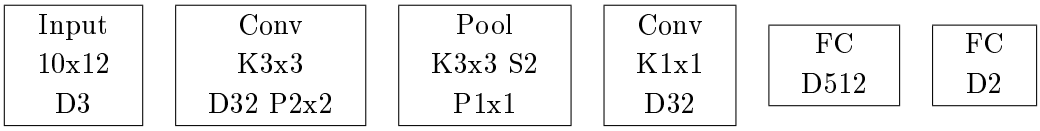
(6) 16x3x3 > 16x1x1 > FC16



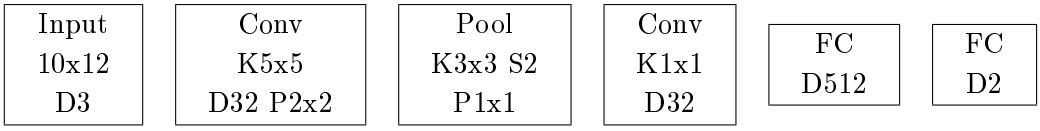
(7) 32x3x2 > 32x1x1 > FC32



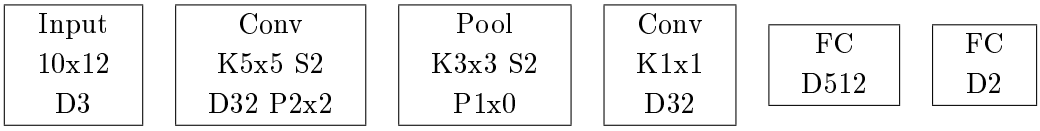
(8) 32x3x2 > 32x1x1 > FC512



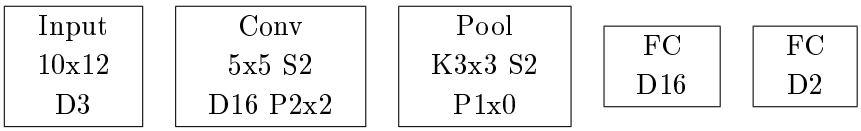
(9) 32x5x5 > 32x1x1 > 512



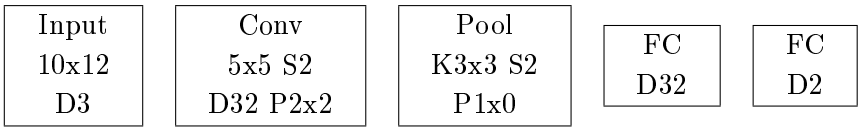
(10) 32x5x5s2 > p3x3s2p1x0 > 32x1x1 > FC512



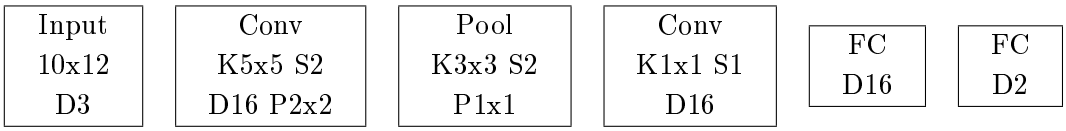
(11) 16x5x5s2 > p3x3s2p1x0 > FC16



(12) 32x5x5s2 > p3x3s2p1x0 > FC32



(13) 16x5x5s2 > 16x1x1 > FC16



(14) 16x5x5s2 > FC16

Input	Conv	Pool	FC	FC
10x12	5x5 S2	K3x3 S2	D16	D2
D3	D16 P2x2	P1x1		

(15) 32x5x5s2 > FC32

Input	Conv	Pool	FC	FC
10x12	5x5 S2	K3x3 S2	D32	D2
D3	D32 P2x2	P1x1		

(16) 16x3x3s2 > FC16

Input	Conv	Pool	FC	FC
10x12	K3x3 S2	K3x3 S2	D16	D2
D3	D16 P1x1	P1x1		

(17) 16x3x3s2 > p3x3s2p1x0 > FC16

Input	Conv	Pool	FC	FC
10x12	K3x3 S2	K3x3 S2	D16	D2
D3	D16 P1x1	P1x0		

Google

Input	Conv	Pool	Conv	FC	FC
13x16	K5x5 S2	K3x3 S2	K1x1 S1	D512	D2
D3	D32 P2x2	P1x1	D32		

5.2.2 16-Network Structures

As the structures of the 12-networks shown earlier, here is the 16-network structures, which are shown to identify their differences. The next chapter will present the performance graphs and the complexity table to be able to compare them.

A list of abbreviations can be seen in Table 5.1 on page 24.

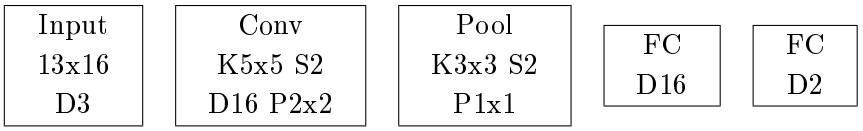
(1) 16x3x3s2 > FC16

Input	Conv	Pool	FC	FC
13x16	K3x3 S2	K3x3 S2	D16	D2
D3	D16 P1x1	P1x1		

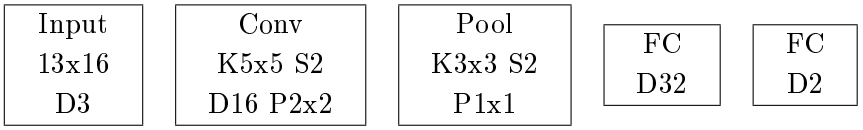
(2) 16x5x5s2 > FC32

Input	Conv	Pool	FC	FC
13x16	K3x3 S2	K3x3 S2	D32	D2
D3	D16 P1x1	P1x1		

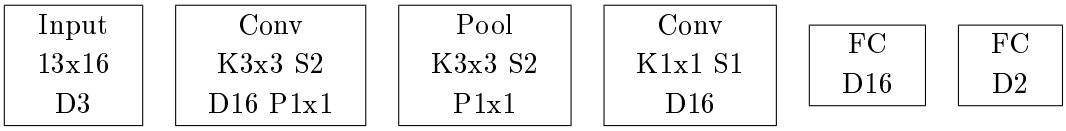
(3) 16x5x5s2 > FC16



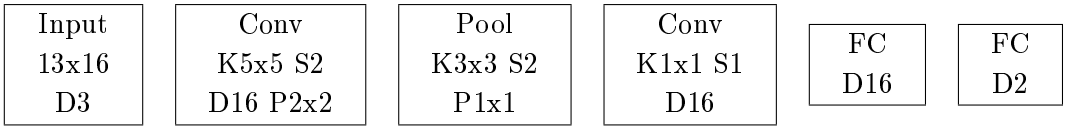
(4) 16x5x5s2 > FC32



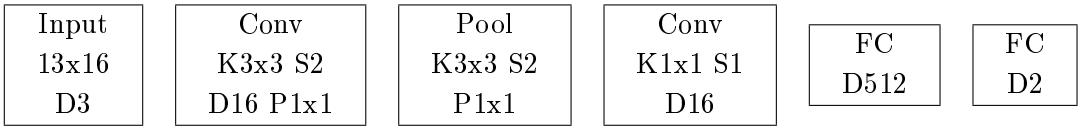
(5) 16x3x3s2 > 16x1x1 > FC16



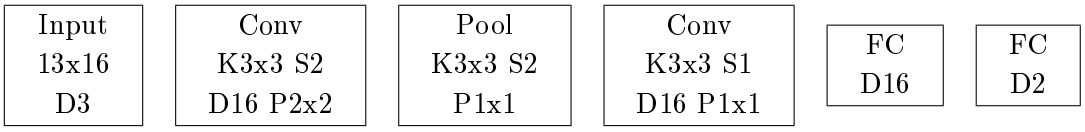
(6) 16x5x5s2 > 16x1x1 > FC16



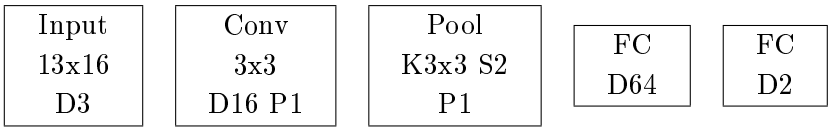
(7) 16x3x3s2 > 16x1x1 > FC512



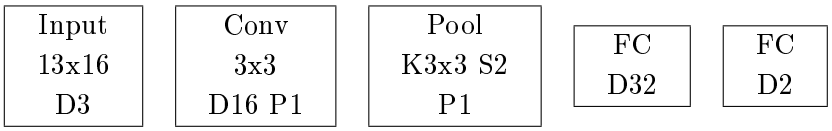
(8) 16x3x3s2 > 16x3x3 > FC16



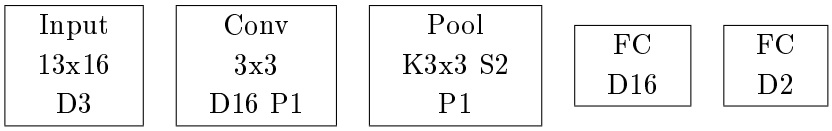
(9) 16x3x3 > FC64



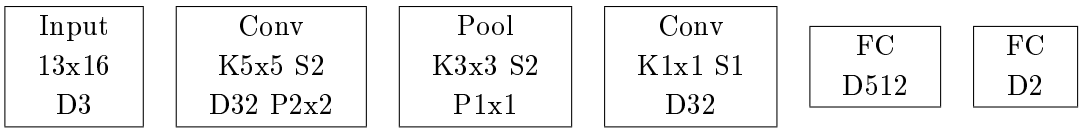
(10) 16x3x3 > FC32



(11) 16x3x3 > FC16



Google



This concludes all the different 12- and 16-network structures, which will be tested in the following chapter to replace the first proposal network from the baseline detector.

Results 6

In this chapter the results of all the proposed networks from the previous chapter will be presented, which will conclude in a final decision for a first network to use in this work's proposed cascade. Finally the performance of the full cascade will be shown and compared to state-of-the-art.

6.1 12-Network Results

Here, the first part of the results of the proposed networks, will be presented. Namely the 12-networks.

6.1.1 12-Network Performance

In order to check network performance, a number of graphs are generated showing important attributes for the first network. These attributes are miss rate, false positives per image, and survived detections windows.

Miss rate as a function of false positives per image is shown on Figure 6.1 on the next page.

It should be noted, that for Figure 6.1 on the following page, the total number of false positives for the Google network, when all detections windows are accepted, is higher than the 12-networks, and therefore, the relation between this network and the others in this graph might not be of any use. By analysing this graph, a trend can be seen, that deeper networks have a higher number of false positives. This could be because the resolution of the image is very small, and therefore, the deeper networks are more overfitted than shallower networks. This is not true for all of the networks, as network 10 has one of the best performances.

Figure 6.2 on the next page shows the relation between miss rate and the threshold of the networks, which at this stage should be very low, because the network will not be able to pick up true detections again. A zoomed graph is shown in Figure 6.3 on page 31. The graphs again show worse performance of deeper networks, except a few, which performs above average (e.g. network 8). It is also seen, that most networks with a stride of two in the convolutional layer do not perform as well as with stride of one.

In Figure 6.4 on page 31, it is seen how many detection windows, which get rejected, as

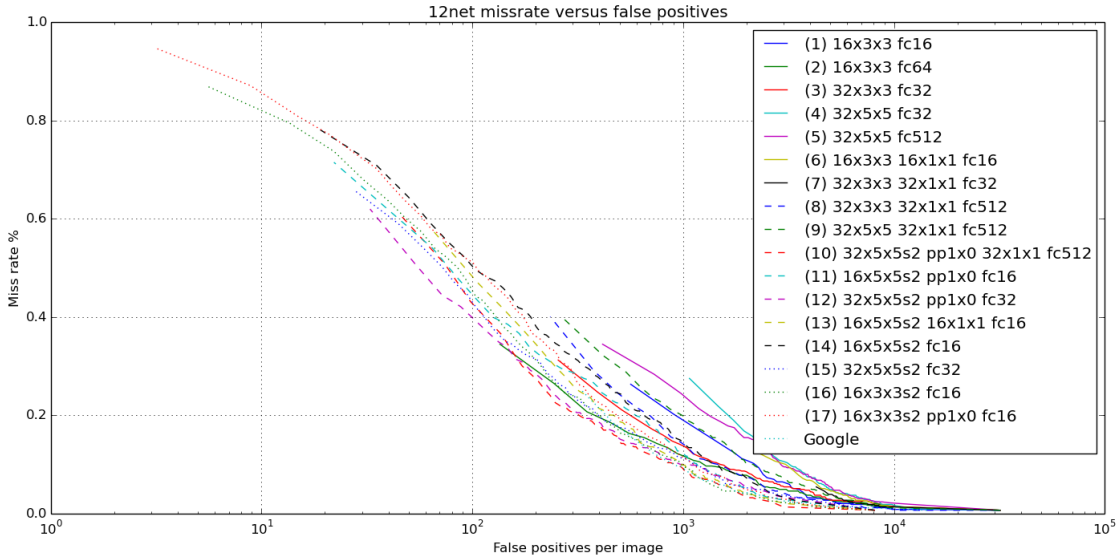


Figure 6.1. Miss rate versus average false positives per image

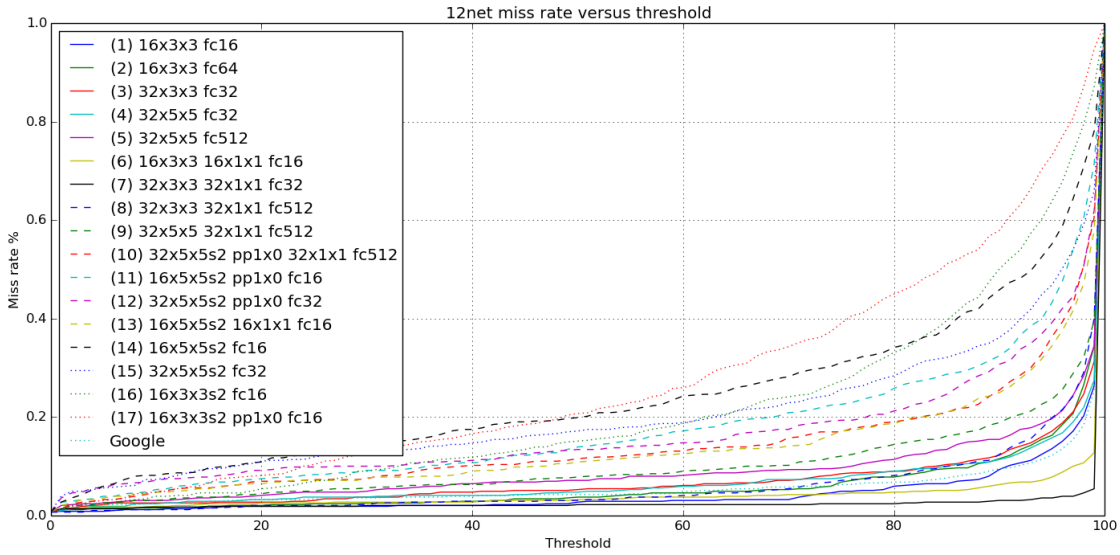


Figure 6.2. Miss rate versus threshold

a function of the threshold of the networks. This is important to know, because the more windows that pass the network means more windows have to be processed by the later, more computationally heavy networks. A zoomed graph of this is shown in Figure 6.5 on page 32. It should be noted from this graph, that the performance resembles the reverse performance from the miss rate against threshold test. Understandingly, networks with a stride of two in the convolutional layer have about a quarter of the survived windows of the ones with a stride of one.

A combination of the two previous described graphs is seen in Figure 6.6 on page 32. It shows the survived windows as a function of miss rate, which is a good way to analyse the performance of one of the earlier networks, as these are the two parameters which

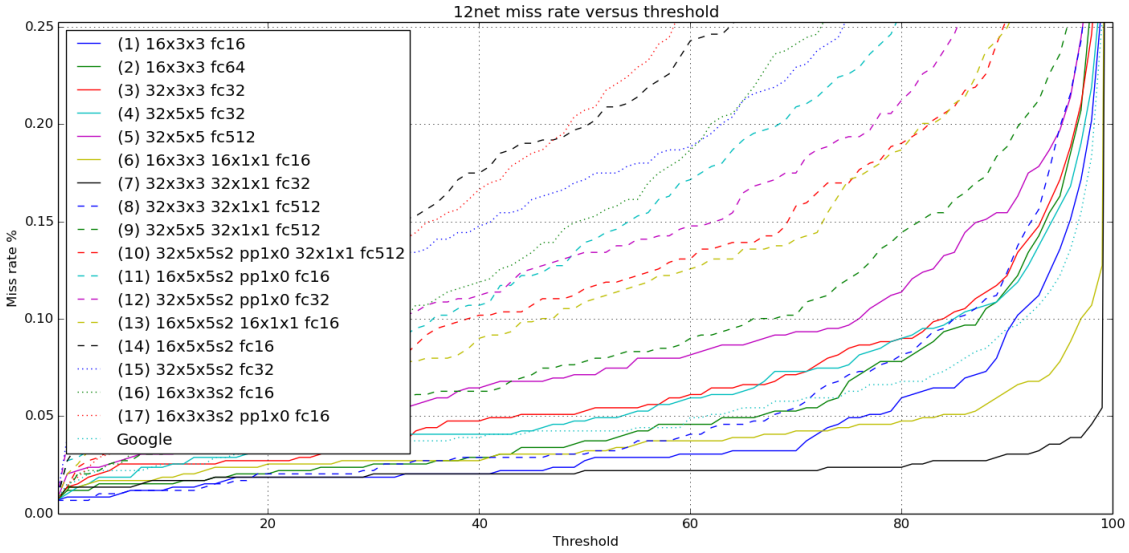


Figure 6.3. Miss rate versus threshold with zoom

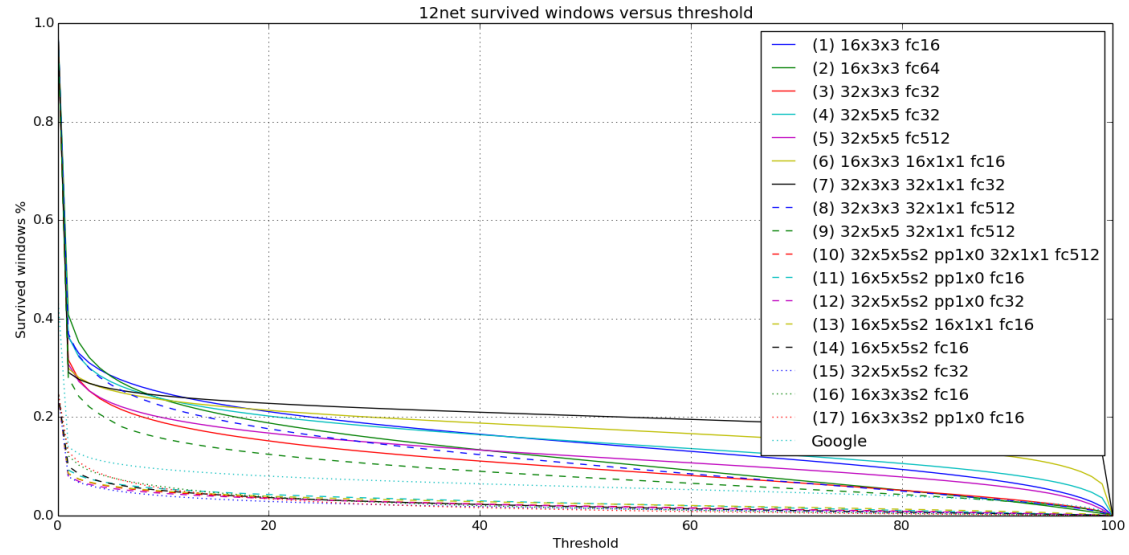


Figure 6.4. Survived windows af the network versus threshold

should be minimal at this point in a cascade. Figure 6.7 on page 33 shows the same figure with zoom. From these graphs, it can be seen that the big reduction in survived windows of the networks with a stride of two makes them outperform the other networks. The best performing network here is a deeper network (network 10), followed by a handful of shallow ones and the google network. The deeper network, is presumably in the top alone, as a deeper network, because it is the only one with a stride of two.

6.1.2 12-Network Complexity

The complexity of the 12-networks can be seen in Table 6.1 on page 33. This table shows the networks with numbers, which corresponds with the number paired with their name

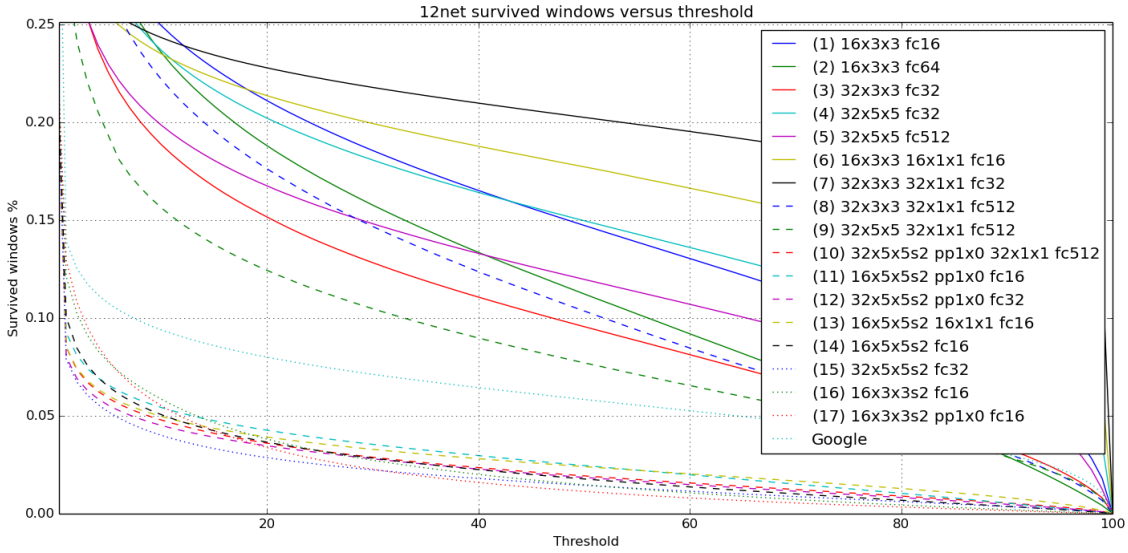


Figure 6.5. Survived windows af the network versus threshold with zoom

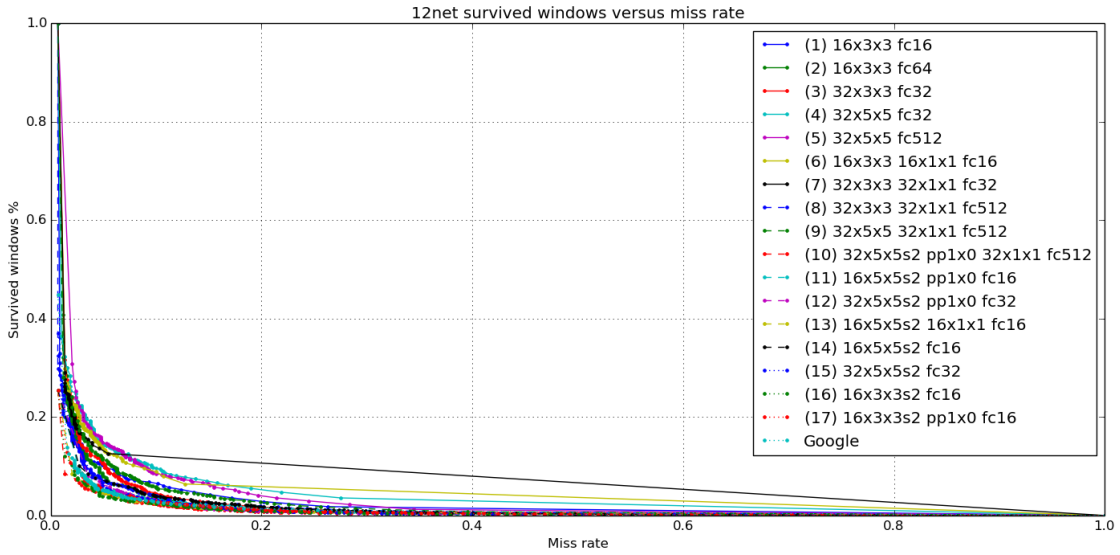


Figure 6.6. Survived windows af the network versus miss rate

when they were presented. The columns show the number of weights, biases, the total number of weights and biases, and the time it takes running the network on the CPU (see CPU specifications on page vi) for each of the networks. The time is calculated by taking 10,000 measurements of the time it takes to forward a single image through the network, and then averaging all the results. Figure 6.8 on page 34 shows the time it takes to forward an image through the network as a function of number of parameters in that network.

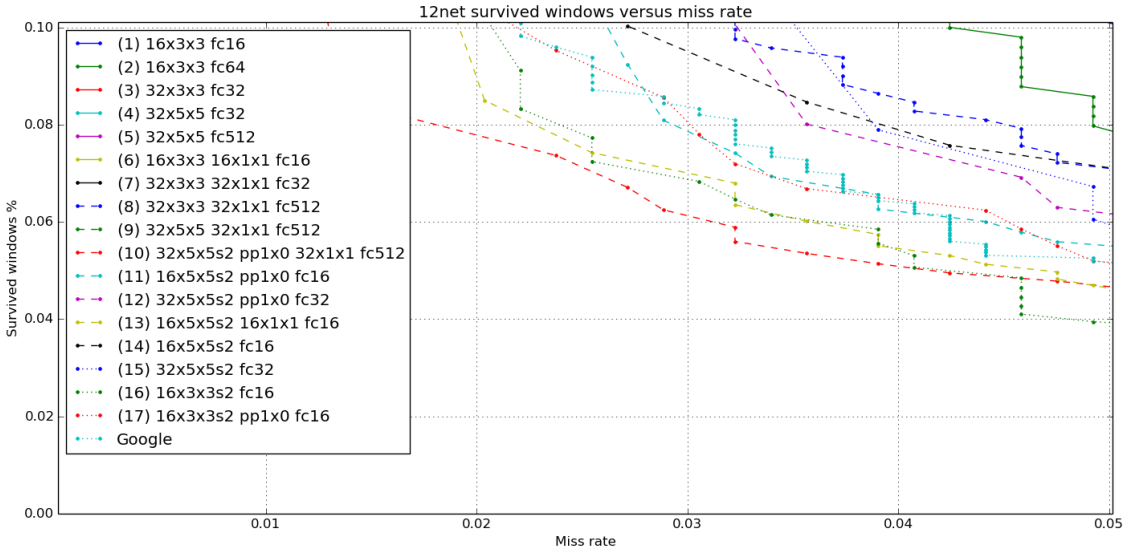


Figure 6.7. Survived windows af the network versus miss rate with zoom

Net	Weights	Bias	Total	Time [μs]
1	11216	34	11250	70.5
2	43568	82	43650	91.4
3	43936	66	44002	100.1
4	45472	66	45538	107.9
5	691552	546	692098	651.8
6	11472	50	11522	73.0
7	44960	98	45058	114.8
8	691040	578	691618	657.7
9	692576	578	693154	673.2
10	151904	578	152482	195.4
11	3536	34	3570	48.5
12	11680	66	11746	60.9
13	4560	50	4610	57.7
14	4304	34	4338	50.0
15	14752	66	14818	64.0
16	3536	34	3570	45.3
17	2768	34	2802	43.9
Google	332128	578	332706	366.4

Table 6.1. The complexity of all the tested 12-networks and the Google network

6.1.3 12-Network Discussion

In summary, it is seen in Figure 6.1 on page 30, that the different networks do not have a huge difference in performance when looking at miss rate against false positives per image. However, by looking at 6.2 on page 30, there is a significant difference, especially at higher thresholds. Furthermore, it should be noted, that the order of the networks, when looking at miss rate, is about the same as the reverse order, when looking at the survived detection windows. From Figure 6.7, it is possible to see the top performing networks in terms of

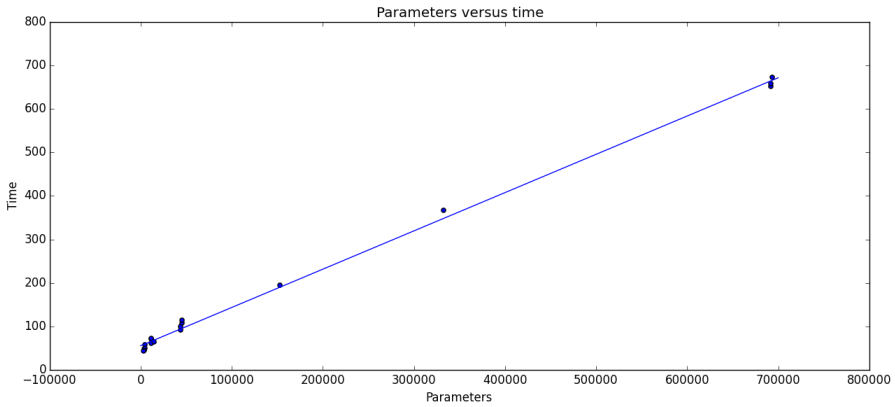


Figure 6.8. Number of parameters versus the computational time for the 12-network

reduction of windows and miss rate, which is network 10, 13, and 16.

From Table 6.1 on the preceding page, it is seen that the size of the fully connected layer is worse at increasing numbers of parameters, and it is also seen that using strides of two reduces the amount of parameters in the network, which happens because the output size before the fully connected is smaller.

From Figure 6.8, it can be seen that on a large scale the time it takes to forward an image through the networks seems to correlate linearly with the number of parameters, though more intermediate data points would be preferred before making that assumption. On a small scale, the data points do not follow the line as well, but this could be because of the variance in time the computer introduces when taking the measurements and the optimisation of the calculations needed (i.e. amount and size of matrix multiplications).

6.2 16-Network Results

Now that all the results of the 12-network testing have been shown, the same performance test of false positives, miss rate and survived windows will be presented for the 16-networks. Not all the same networks from the previous section will be tested. Only a subset of the better performing ones, because it is assumed that the relation of the networks does not change all that much with this small increase in size.

6.2.1 16-Network Performance

Firstly in Figure 6.9 on the facing page the miss rate versus the average false positives per image is seen. This graph shows, that what was a tiny decreasement in the 12-networks increased to significantly underperforming networks when the strides of the convolutional layer is one. Other than that the performance only differs a little bit between the networks.

Figure 6.10 on the next page shows the miss rate versus the threshold of the networks, and Figure 6.11 on page 36 shows a zoomed version of the same graph. Here it can be seen, that the networks with a stride of one, outperforms the ones with a stride of two, and that it is possible to group the networks in three ”performance groups”, as there are gaps between them.

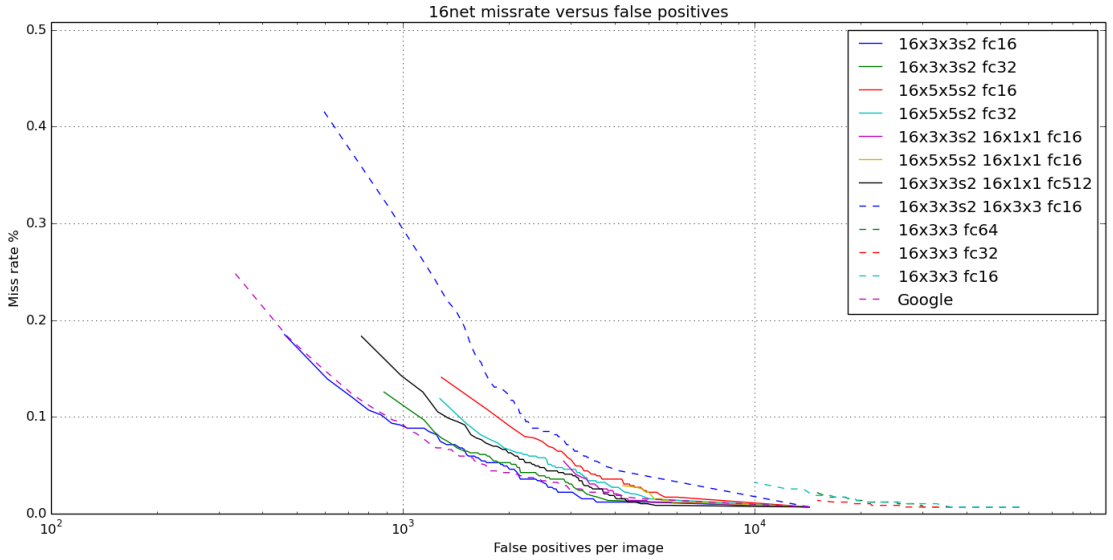


Figure 6.9. Miss rate versus average false positives per image

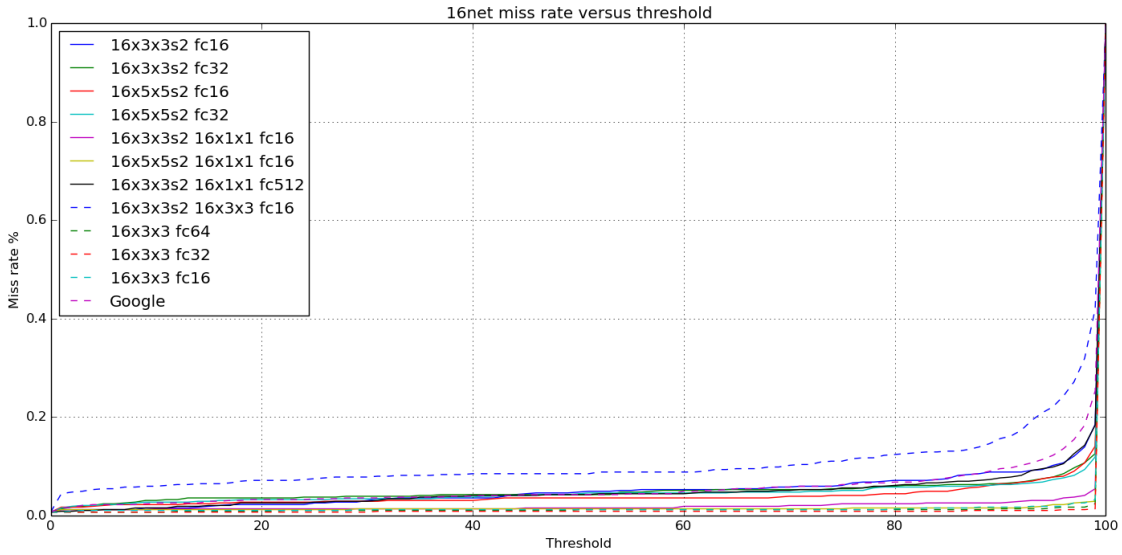


Figure 6.10. Miss rate versus threshold

The number of survived detection windows after the network versus the threshold of the network, is seen in Figure 6.12 and Figure 6.13 on page 37 shows a zoomed version of this. From these graphs, it is seen again, that increasing the stride to two, significantly decrease the number of survived windows, even more than for the 12-networks. The networks also still cluster into groups, but not the same groups as in the miss rate case.

The final graph, showing survived windows versus miss rate of the 16-networks, can be seen on Figure 6.14 on page 37 and 6.15 on page 38 for zoom. Here it is apparent, that the networks with a stride of one in the convolutional layer performs worse because of the high number of survived windows. Apart from that there is no distinct network type which outperforms the rest, because both shallow and deeper networks are among the top

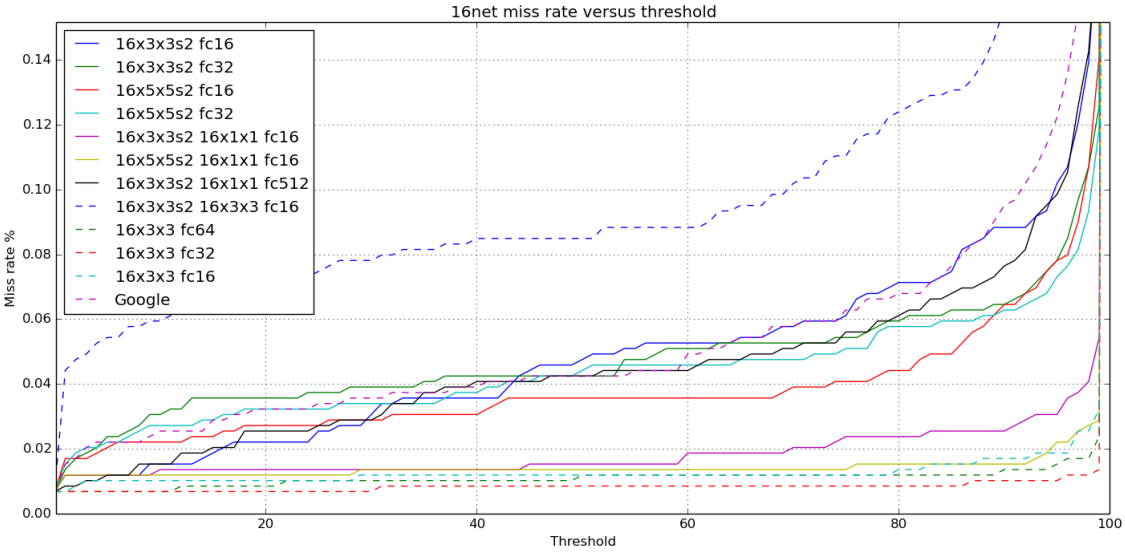


Figure 6.11. Miss rate versus threshold with zoom

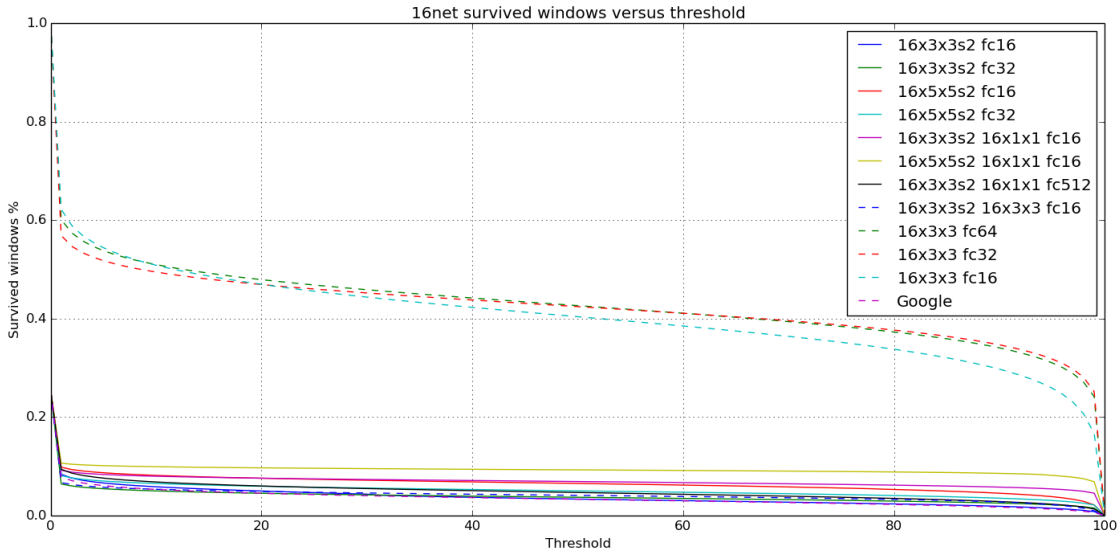


Figure 6.12. Survived windows of the network versus threshold

performing networks. The top performing network is network 1.

6.2.2 16-Network Complexity

In order to know the complexity of the 16-networks, a table like for the 12-networks is presented and can be seen in Table 6.2 on page 38.

As for the 12-network, Figure 6.16 on page 38 shows the time it takes to forward in image through the 16-network as a function of number of parameters in each network. However for the 16-networks, there is not clear relation between the time and the number of parameters.

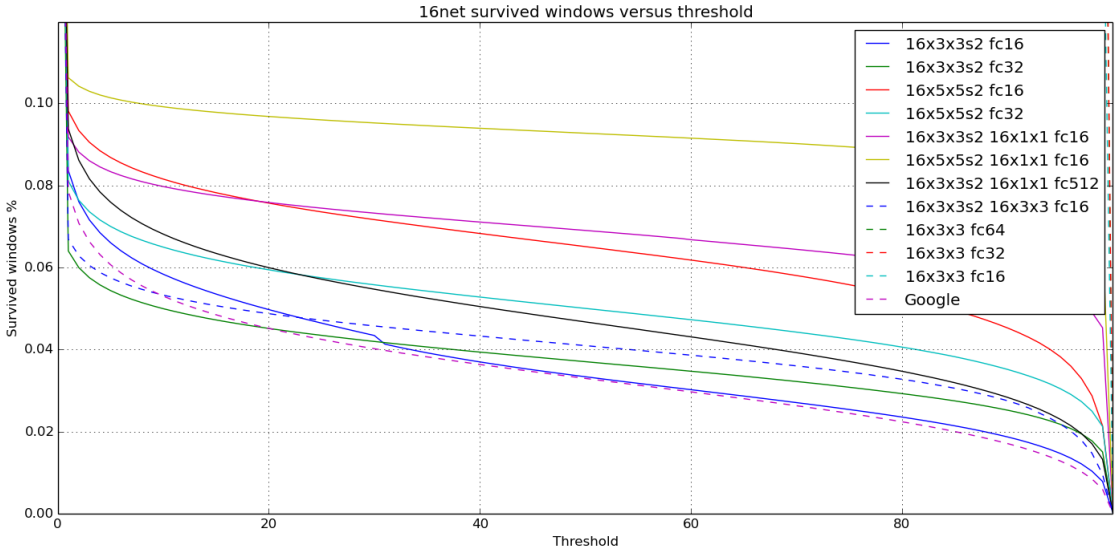


Figure 6.13. Survived windows of the network versus threshold with zoom

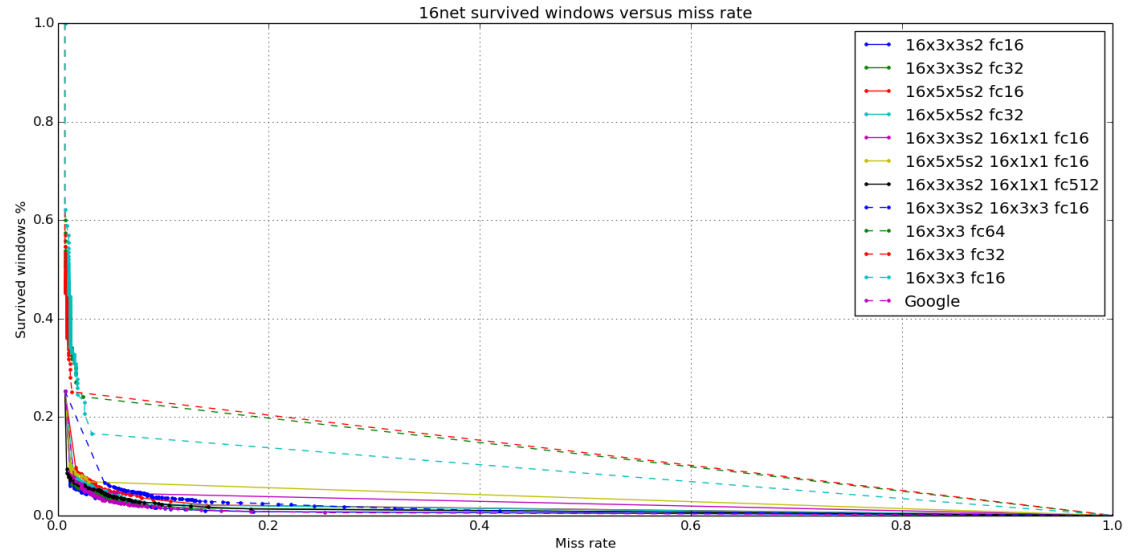


Figure 6.14. Survived windows of the network versus miss rate

6.2.3 16-network Discussion

By looking at Figure 6.11 on the facing page and 6.13, as for the 12-networks, the order of the performance is reversed. But some networks stand out with good performances in miss rate, but get a mediocre result in survived detection windows, or with good in survived detection windows, but mediocre in miss rate, compared to the others 16-networks. These networks can be seen better in Figure 6.15 on the following page, which shows the top performing networks for the 16-network as network 1, 2, and 12.

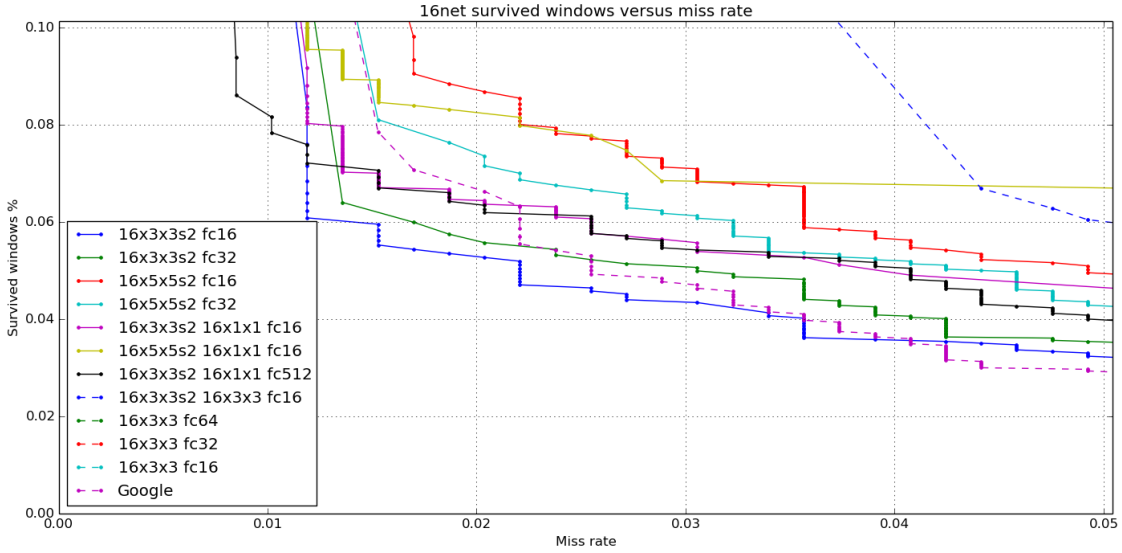


Figure 6.15. Survived windows of the network versus miss rate with zoom

Net	Weights	Bias	Total	Time [μs]
1	5584	34	5618	55.3
2	10736	50	10786	60.5
3	6352	34	62.1	62.1
4	11504	50	11554	67.8
5	5840	50	5890	64.5
6	6608	50	6658	71.3
7	165552	546	166098	213.4
8	7888	50	7938	71.5
9	82480	82	82562	226.7
10	41456	50	41506	216.3
11	20944	34	20978	205.4
Google	332128	578	332706	371.4

Table 6.2. The complexity of all the tested 16-networks

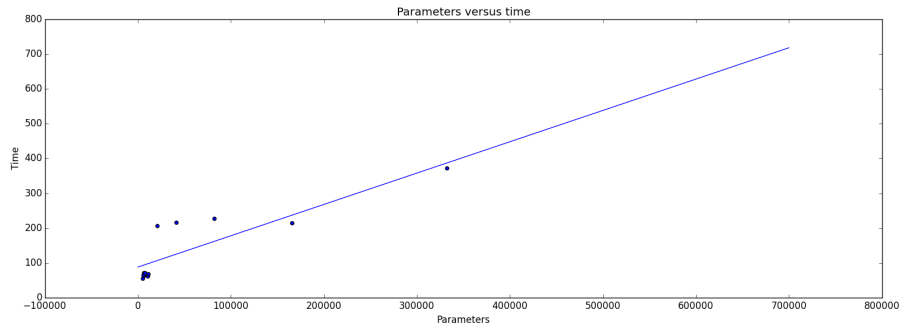


Figure 6.16. Number of parameters versus the computational time for the 16-networks

6.2.4 First Network Discussion

As said previously, there are some things, which are important to take into account, when choosing a first network. As seen from the graphs from the 12-networks and the 16-networks, the general difference between the networks is that the 12-networks have higher miss rate than the 16-networks, but they also reject more detection windows.

The time it takes to run a detection window through the 16-networks is about 23% slower than it takes to run through a similar 12-network, for shallow networks, but it seems to be exponentially increasing with more complex networks, as the most complex one is about 90% slower as a 16-network then a 12-network.

It is chosen, that a reduction in survived windows of more than 90% is wanted from the network, as well as a miss rate lower than 5%. Therefore, it is also taken into account, how stable the performance is within these limits (i.e. how big a part of the all threshold is within this limit). As seen on Figure 6.7 on page 33 and Figure 6.15 on the facing page, each point on the curves represent one percent point change in threshold.

It is chosen to use a 16-network, because the increase in recall is thought to make up for the complexity increase, and the increase in number of detection windows. To get back on these speed decreasements, a shallow 16-network will be chosen. This is to be in the low end in number of detection windows and complexity, but at the expense of some accuracy. Specifically the 16x3x3s2>FC16 network from section 5.2.2 on page 26.

6.3 First Network Comparison

Now that a new first proposal network has been chosen, it is compared in performance with the old network, to see if it provides an increase or decrease in performance, which can be seen in Figure 6.17 on the following page. From this graph, it can be seen that it does not perform as well as the baseline's first network, but it is expected to have some reduction in performance, as it is only looking at upper body, which is a subset of a full body and there are less features to learn from. This has about 10 to 100 times more false positives per image (FPPI) than the baseline.

6.4 Comparing Against State-of-the-Art

After this comparison of the first network in the cascades, the full cascade network is tested and with the results shown in Figure 6.18 on the next page. It is compared with Doppia [Benenson et al., 2012].

Examples of images from some of the detectors shown in Figure 6.18 on the following page are seen in figures from Figure 6.19 on page 41 to Figure 6.27 on page 42.

6.5 State-of-the-Art Discussion

As seen in Figure 6.18 on the next page, the results for upper body detection does not perform as well as full-sized pedestrian detectors. The reason behind this could be, as stated in the comparison of the first network in the cascade, that this detector only has

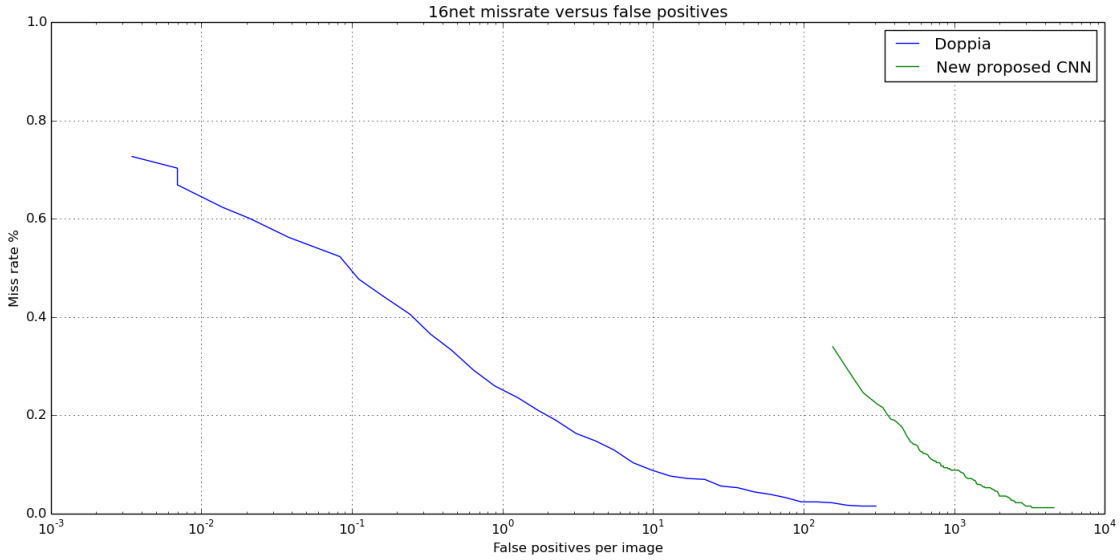


Figure 6.17. A comparison between the performance of the baseline cascade's first network and the new propose network

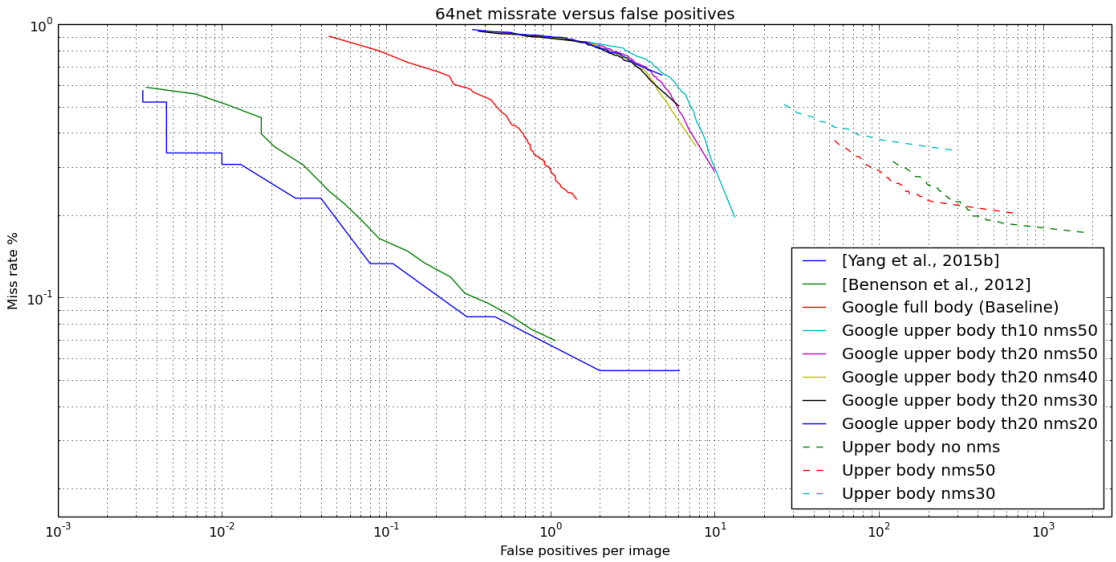


Figure 6.18. The results of the complete cascade done on the INRIA test dataset

the features for the upper bodies and not all from full-sized bodies, as upper bodies are a sub-part of full-sized pedestrians.

From the images in Figure 6.19 on the facing page to Figure 6.24 on the next page, it is seen how the detection windows change from full-sized to upper body. The main difference is, that the detector is much worse at differentiating the windows. This is seen by vast numbers of windows with a confidence score of '1.00'. This could mean, the detector does not have a clear enough definition of an upper body, likely by not having good features for defining what an upper body is. It does however, also have good detections as seen in Figure 6.27 on page 42.

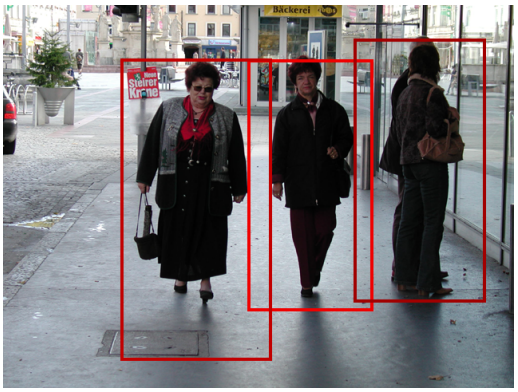


Figure 6.19. Example image, from INRIA, with the detector by Benenson et al. [2012]

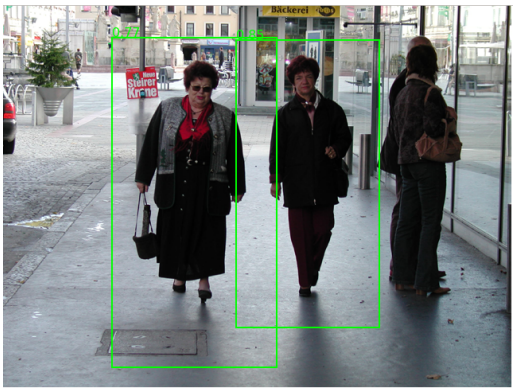


Figure 6.20. Example image, from INRIA, with the basline detector by Angelova et al. [2015]

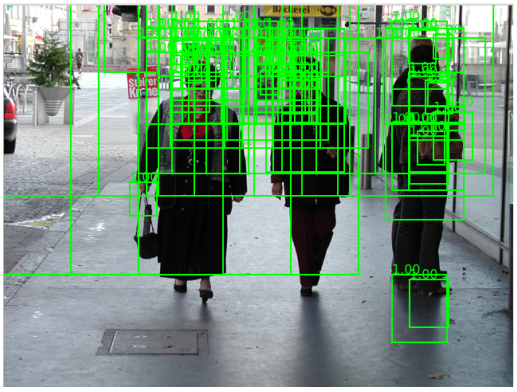


Figure 6.21. Example image, from INRIA, with the proposed upper body detector

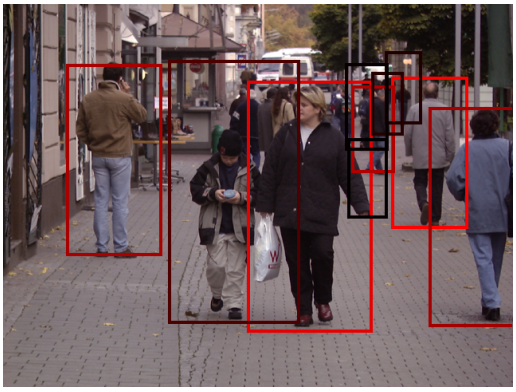


Figure 6.22. Example image, from INRIA, with the detector by Benenson et al. [2012]

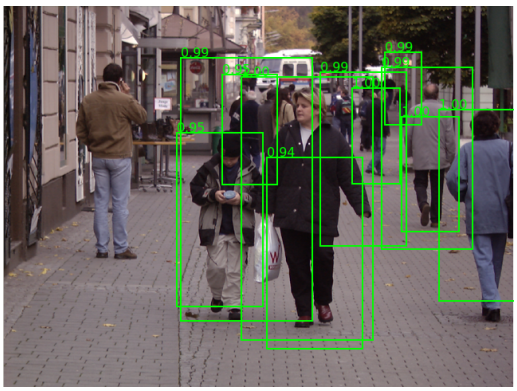


Figure 6.23. Example image, from INRIA, with the basline detector by Angelova et al. [2015]

Figure 6.24. Example image, from INRIA, with the proposed upper body detector

These results can also be interpreted as an indication of how much pedestrian detectors use lower body features for detection.

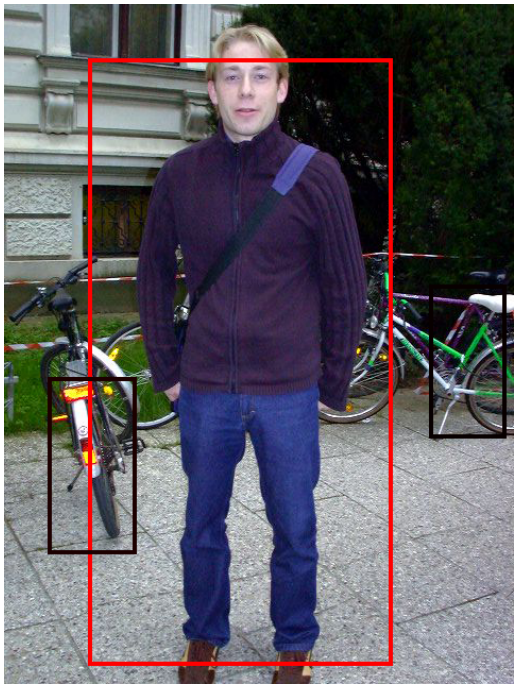


Figure 6.25. Example image, from INRIA, with the detector by Benenson et al. [2012]

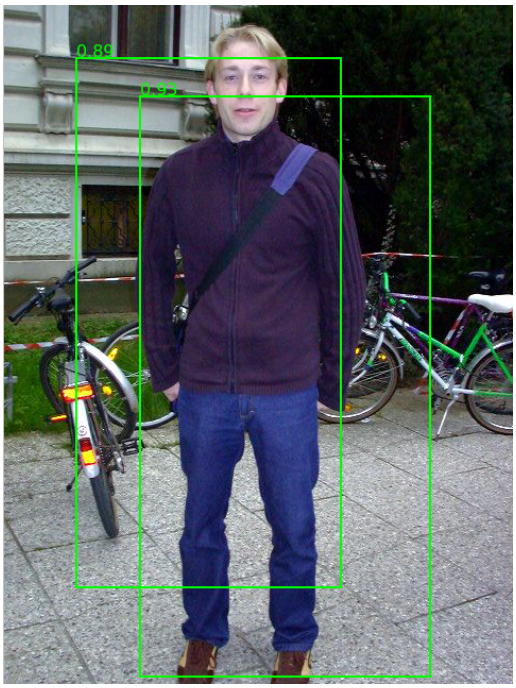


Figure 6.26. Example image, from INRIA, with the basline detector by Angelova et al. [2015]

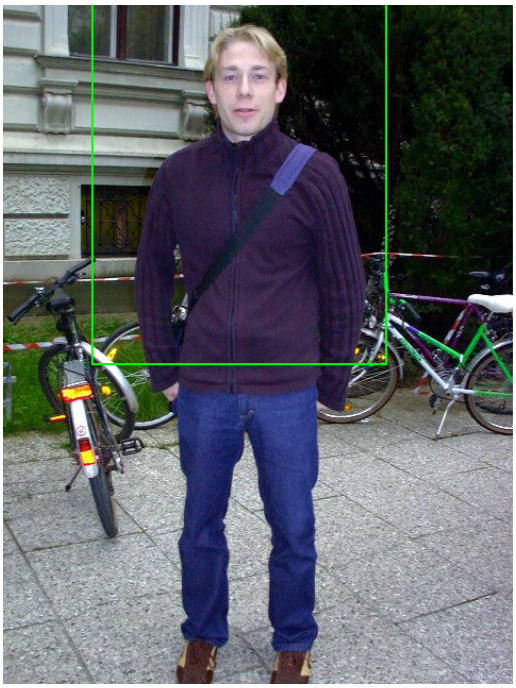


Figure 6.27. Example image, from INRIA, with the proposed upper body detector

Evaluation 7

In this chapter a conclusion of the project according to the problem statement will be presented along with what could be done for future work to improve upon this project.

7.1 Conclusion

The initiating problem statement is the prerequisite for the conclusion of this project, which is the following:

> How is it possible to design a system for pedestrian detection,
> which achieves good performance,
> as well as handles segmentation of pedestrians' lower body?

A system for pedestrian detection, which handles segmentation of pedestrians' lower body, is designed, but this system is not able the achieve good performance. This can be seen in Figure 6.18 on page 40, that when more of the system was converted to only look at upper bodies, the worse it becomes.

It is seen, that changing from a full-sized, upper body, upper body cascade into a upper body, upper body, upper body cascade, the performance is decreasing, which is peculiar as the final deep network remains the same and should have the same performing capabilities as before. The reason for this might be that the proposed first network's hard negatives have more variance than the ones from the baseline's first network. Therefore, this variance travels all the way up through the cascade and into the last network, which at this point will have more varying data, and therefore, not be as specialised at the cost of performance.

The reason behind these bad results point to, that the upper body lacks the features needed for proper performance, with this specific detector's architecture or that the proposed network can not learn the features needed to distinguish properly between upper bodies and non upper bodies. Given another network structure it might be possible to design a well performing upper body detector, but presumably still with reduced performance compared to full-sized detectors, but improving upon the results presented in this project.

A reason why the proposed network is having trouble scoring detections windows could be because upper bodies do not have a clear contour at the bottom of the upper body. The

transition between torso and hip can not always be clearly seen, as it could be obscured by a jacket or a long shirt. The contour of a full-sized pedestrian would be visible at all times except in the case when background and pedestrian are similarly coloured, or in case of occlusion, but these problem also exists for upper bodies.

7.2 Future Work

For future work, a better baseline implementation with state-of-the-art performance would be advantageous, as it would diminish the likelihood of mistakes made for training the networks, which could couse a reduction in performance.

From the network structure's point of view, it could be that upper body detection needs be to implemented with better suited features, to solve this detection problem. It might be that the size of the network is too small and that an upper body detector needs a deeper network, in order to construct more complex features, for better classification.

Better feature extraction might also increase performance, which could be attained by using a pre-trained model for general object detection to have a broader range of features and not just the features trained to make the split between upper body and background patches.

Another way to improve performance, could be to change the structure of the network completely and use a part based model instead. This will then be looking at the parts present in upper bodies like head, torso, and arms as it is also done for full-sized pedestrians, but again only in a sub-part of a pedestrian.

It might also be that using CNN is the wrong choice, as it might be a harder problem with this type of network than it would by using a classical approach like HOG or Haar-like features.

Bibliography

Angelova, Krizhevsky, Vanhoucke, Ogale, and Ferguson, **2015**. Anelia Angelova, Alex Krizhevsky, Vincent Vanhoucke, Abhijit Ogale, and Dave Ferguson. *Real-Time Pedestrian Detection With Deep Network Cascades*. 2015. URL `http://www.vision.caltech.edu/anelia/publications/Angelova15RealTimePedestrian.pdf`.

Benenson, Mathias, Timofte, and Gool, **2012**. Rodrigo Benenson, Markus Mathias, Radu Timofte, and Luc Van Gool. *Pedestrian detection at 100 frames per second*. 2012. URL `http://rodrigob.github.io/documents/2012_cvpr_pedestrian_detection_at_100_frames_per_second.pdf`.

Navneet Dalal and Bill Triggs, June 2005. Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005. URL `http://lear.inrialpes.fr/pubs/2005/DT05`.

Danmarks Statistik, **2014a**. Danmarks Statistik. *FÃ¦rdselsuheld*. 2014. URL `https://www.dst.dk/da/Statistik/emner/trafikulykker/faerdselsuheld`.

Danmarks Statistik, **2016**. Danmarks Statistik. *Fem gange sÃ¥ mange personbiler som i 1961*. 2016. URL `http://www.dst.dk/da/Statistik/NytHtml?cid=21503`.

Danmarks Statistik, **2014b**. Danmarks Statistik. *Trafik*. 2014. URL `https://www.dst.dk/da/Statistik/emner/transport/trafik`.

Dollár, Wojek, Schiele, and Perona, **2012**. Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. *Pedestrian Detection: An Evaluation of the State of the Art*. PAMI, 34, 2012.

A. Ess, B. Leibe, K. Schindler, , and L. van Gool, June 2008. A. Ess, B. Leibe, K. Schindler, , and L. van Gool. A Mobile Vision System for Robust Multi-Person Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*. IEEE Press, June 2008.

Everingham, Van Gool, Williams, Winn, and Zisserman. M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

Farfade, Saberian, and Li, **2015**. Sachin Sudhakar Farfade, Mohammad Saberian, and Li-Jia Li. *Multi-view Face Detection Using Deep Convolutional Neural Networks*. 2015. URL http://arxiv.org/pdf/1502.02766v3.pdf.

Fukui, Yamashita, Yamauchi, Fujiyoshi, and Murase, **2015**. Hiroshi Fukui, Takayoshi Yamashita, Yuji Yamauchi, Hironobu Fujiyoshi, and Hiroshi Murase. *Pedestrian Detection Based on Deep Convolutional Neural Networks With Ensemble Interface Network*. 2015. URL http://www.vision.cs.chubu.ac.jp/MPRG/C_group/C070_fukui2015.pdf.

González, Villalonga, Xu, Vázquez, Amores, and López, **2015**. Alejandro González, Gabriel Villalonga, Jiaolong Xu, David Vázquez, Jaume Amores, and Antonio M. López. *Multiview Random Forest of Local Experts Combining RGB and LIDAR data for Pedestrian Detection*. 2015. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7225711.

Karpathy. Andrej Karpathy. *Convolutional Neural Networks for Visual Recognition*. URL http://cs231n.github.io/convolutional-networks/.

Krizhevsky, Sutskever, and Hinton, **2012**. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Li, Liang, Shen, Xu, and Yan, **2015**. Jianan Li, Xiaodan Liang, ShengMei Shen, Tingfa Xu, and Shuicheng Yan. *Scale-aware Fast R-CNN for Pedestrian Detection*. 2015. URL http://arxiv.org/pdf/1510.08160.pdf.

Lit, Lin, Shen, Brandt, and Huat, **2015**. Haoxiang Lit, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Huat. *A convolutional Neural Network Cascade for Face Detection*. Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on, pages 5325 – 5334, 2015. URL http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Li_A_Convolutional_Neural_2015_CVPR_paper.pdf.

Ouyang and Wang, **2013**. Wanli Ouyang and Xiaogang Wang. *Joint Deep Learning for Pedestrian Detection*. The IEEE International Conference on Computer Vision (ICCV), pages 2056–2063, 2013. URL http://www.cv-foundation.org/openaccess/content_iccv_2013/papers/Ouyang_Joint_Deep_Learning_2013_ICCV_paper.pdf.

Ruder, **2016**. Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2016. URL http://sebastianruder.com/optimizing-gradient-descent/.

Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov, **2014**. Nitish Srivastava, Geo rey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. 2014. URL https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf.

Tian, Luo, Wang, and Tang, **2015**. Yonglong Tian, Ping Luo, Xiaogang Wang, and Xiaoou Tang. *Pedestrian Detection aided by Deep Learning Semantic Tasks*. Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on, 2015. URL `http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Tian_Pedestrian_Detection_Aided_2015_CVPR_paper.pdf`.

Yang, Yan, Lei, and Li, **2015a**. Bin Yang, Junjie Yan, Zhen Lei, and Stan Z. Li. *Convolutional Channel Features*. 2015. URL `http://www.cbsr.ia.ac.cn/users/byang/papers/ccf.pdf`.

Yang, Wang, and Wu, **2015b**. Yi Yang, Zhenhua Wang, and Fuchao Wu. *Exploring Prior Knowledge for Pedestrian Detection*. 2015. URL `http://www.bmva.org/bmvc/2015/papers/paper176/paper176.pdf`.

Zhang, Benenson, and Schiele, **2015**. Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. *Filtered Channel Features for Pedestrian Detection*. 2015. URL `http://arxiv.org/pdf/1501.05759v1.pdf`.