Failure Detection in Pedestrian Detection : A Fully Convolutional Neural Network approach

VGIS10 - Master Thesis Report

Christos Apostolopoulos

Aalborg University
Electronics and IT

Electronics and IT
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Failure Detection in Pedestrian Detection : A Fully Convolutional Neural Network approach

Theme:
Master's Thesis: Vision, Graphics, and Interactive Systems

Project Period:
Fall Semester 2015/Spring Semester 2016

Project Group:
VGIS 1047

Participant(s):
Christos Apostolopoulos

Supervisor(s):
Kamal Nasrollahi
Ming-Hsuan Yang

Copies: 1

Page Numbers: 75

Date of Completion:
June 2, 2016

Abstract:

Pedestrian detection forms a key problem in computer vision, with applications that can greatly affect every day life. In this project, failures in pedestrian detectors are attempted to be refined by re-evaluating their results via a fully convolutional neural network. The network is trained on a number of datasets which include a custom partial occluded pedestrian dataset.
The networks efficiency was evaluated by examining the loss through training (iterations) and by measuring the mean interestion union across classes. As a detector, accuracy was measured by comparing the network given a detectors result against state-of-the-art but also by measuring the networks refined result against the detector without the network.
It was found that although results vary, the proposed network shows promising results.

Contents

Preface **vii**

1 Introduction **1**
1.1 Pedestrian Detection . . . 2
1.2 Failures in pedestrian detection . . . 2

2 Initial Problem Statement **5**
2.1 Pedestrian Failure Detection Methods . . . 5
2.1.1 Input Image . . . 5
2.1.2 Feature Extraction . . . 6
2.1.3 Feature Representation . . . 8
2.1.4 Pedestrian Detection . . . 9
2.1.5 Refined Result . . . 10

3 Related Work **11**
3.1 Hand-crafted features . . . 11
3.2 Convolutional Neural Networks . . . 15

4 Final Problem Statement **29**
4.1 Problem statement . . . 29
4.2 Limitations . . . 29
4.3 Requirements . . . 30
4.4 System Design . . . 30
4.4.1 Offline . . . 31
4.4.2 Online . . . 32
4.5 Datasets . . . 32

5 Design **37**
5.1 Offline . . . 37
5.1.1 Architecture . . . 37
5.1.2 Resize/Distort data . . . 41
5.1.3 Augment Data . . . 43

5.1.4 Network Training Parameters 45
5.2 Online 46
5.2.1 Extract bounding box coordinates 46
5.2.2 Apply scaling factor 47
5.2.3 Pre-process inputs/Pass through network 47
5.2.4 Calculate new bounding box coordinates/Re-adjust result . . 48

6 Results **49**
6.1 Offline 49
6.2 Online 52
6.2.1 Comparison with State-of-the-art 53
6.2.2 Segmentation Error/Scaling Factor Test 58
6.2.3 Analysis of Results 61

7 Conclusion **67**

A Labeling program **69**

Bibliography **71**

Preface

This report contains the work of a long master thesis in the master program of Aalborg University in Vision, Graphics and Interactive Systems starting September 2015 and ending June 2016. The majority of this master thesis was written in University of California, Merced (UCM) where the author collaborated with the Computer Vision Lab team.

During this report several external sources will be used, which will be referred to when used. The reference is seen as a number in square brackets like this: [#]. The number refers to the source in the bibliography, which is found at the end of this report and is generated automatically by bibtex. In the bibliography the books are described with its author, title, pages, publishers, edition, and year or author, title and URL. Figures and tables are numbered according to which chapter they appear in, i.e. the first figure in Chapter 7 has the number 7.1, the second figure has number 7.2 etc. Abbreviations are found in the word list on page vii. Appendices are found at the end of the report, and are referred to with A, B, C, ...

Aalborg University, June 2, 2016

Christos Apostolopoulos
<capost13@student.aau.dk>

Acknowledgments

I would like to deeply thank AAU Associate Professor Kamal Nasrollahi for his patient guidance, constant availability and efficient supervising. I deeply thank UCM Associate Professor Ming-Hsuan Yang for giving me the chance of having this abroad experience and being able to work on such a high research level and for his responsible and efficient supervising.

Furthermore, I would like to thank all the PhD candidates that offered me help, advice and collaboration throughout this master thesis: Yi-Hsuan Tsai (UCM), Wei-Sheng Lai (UCM), Yijun Li (UCM), Guangyu Zhong (Dalian University of Technology), Jia-Bin Huang (University of Illinois at Urbana-Champaign), Chao Ma (Shanghai Jiao Tong University) and especially Sifei Liu (UCM).

Chapter 1

Introduction

Object detection is a fundamental and important area of computer vision with many applications such as surveillance systems, human computer interaction and auto-control systems, where an object of interest is attempted to be located or detected over a series of frames/images. Due to the importance of its applications, it has gained much attention over the years and many different algorithms have been developed in order to solve the problem.

The goal in object detection is to be able to robustly detect an object under varying situations such as camera distances, angles, object perturbations, weather conditions. Approaches usually deal exclusively with the object presented (in terms of features) and how it can be tracked (an example in pedestrian detection can be seen at Figure 1.1). Typically, the most important resulting factors in a detection system include:

- The position of the object (x,y coordinates)
- The size of the object (x,y coordinates, width and height)
- A score which denotes how confidence the system is about the result being a pedestrian

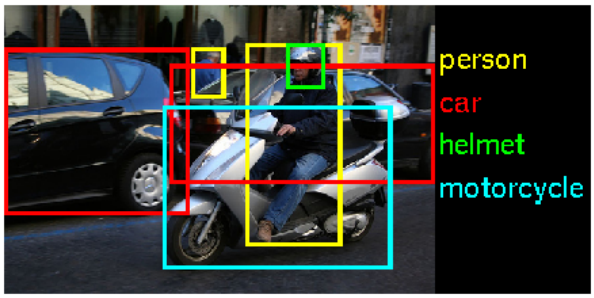

Figure 1.1: Output of object detector. Each calculated bounding box is associated with a specific class ([37]).

1.1 Pedestrian Detection

One of the most common detection objects of interest include pedestrians, where in a street with unknown situations (with respect to challeging factors) a single or multiple pedestrians are attempted to be detected. Over the years, techniques such as moving vehicles, thermal cameras, or simple stationary cameras have been used for dealing with this. Typical applications include monitoring pedestrian routes for shopping preferences, or security reasons.

Figure 1.2: Left: Pedestrian Detection using moving vehicles ([20]). Right: Detecting, tracking and counting pedestrians using stereo thermal cameras ([28]).

1.2 Failures in pedestrian detection

Even though algorithms have been succefully implemented over the past years for pedestrian detection, there is a research gap relative to failures in the algorithms due to various challenging factors. When a detector fails to enclose a pedestrian correctly there is no way to recover and therefore will just output the incorrect result. Implementing a system which is able to recognize these failures effectively and recover (if possible) the detectors estimation would increase accuracy and precision by a large margin.

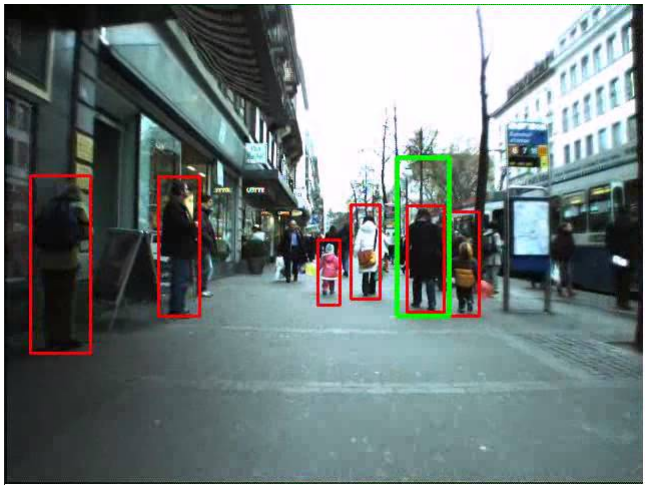

Figure 1.3: An example where a given detector fails to correctly detect the pedestrian of interest. Green box denotes detector result where red box denotes ground truth ([14]).

Failures are most commonly associated with challenges which correspond to real-life scenarios. For example, in one image/video the scale of the object may be miscalculated, the object may be occluded heavily or it may suffer from background clutter (Figure 1.4).

Figure 1.4: Various challenging factors. From left to right: Low resolution ([7]), Occlusion ([48]), Scale ([49]).

Chapter 2

Initial Problem Statement

The following chapter focuses on initializing the problem given from the introduction and analyzes the ways it can be dealt with.

2.1 Pedestrian Failure Detection Methods

Detecting and fixing failures in detection results is a topic that is not that explored in the area of computer vision. In object tracking, there has been an attempt to detect failures ([25]). Detectors usually trust a confidence score (a constant which denotes how confident the algorithm is that the bounding box returned contains a pedestrian) returned without further refining the given result. That being said, it is not straightforward to define what a failure detection method is consisted of. Nevertheless, an intuitive pipeline can be shown at Figure 2.1:

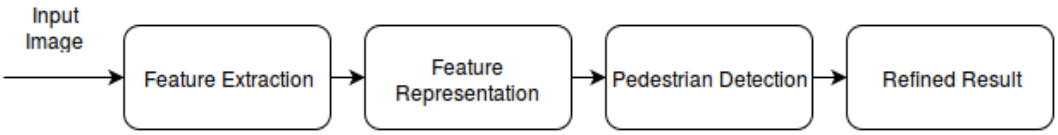

Figure 2.1: Pedestrian Failure Detection Pipeline

2.1.1 Input Image

The input image is the basis of the processing and detection that follows in the pipeline. Since the area of interest is detection, the input is expected to be a bounding box cropped from an image(most typically, from a street) which should include a pedestrian in some region in order to be detected and the result to be refined (Figure 2.2).

Figure 2.2: Input RGB Image Examples [49], [29]

Different input images consist of different properties. These properties mainly have to do with a number of the challenging scenarios mentioned in Section 1.2. It is unknown at this stage whether the result is a good approximation or whether the input contains a large amount of background. For example in Figure 2.2, the right example contains an enclosed pedestrian, the left example contains a large amount of background(scale issue) and the middle one contains a partialy occluded pedestrian at a low resolution. Furthermore, the input may not even contain a pedestrian (failure of algorithm - false positive).

Even though the input is not in the systems control, it is significant to build a system which can robustly handle these variations in which appear a number of different factors.

The input in many cases can be pre-processed for the systems need, meaning algorithms such as background subtraction and image resizing can be applied. These can further contribute into the problem of the unknown input and the efficient detection and result refinement.

2.1.2 Feature Extraction

Given the input image, a decision regarding the features that will be used for representing the input needs to be made. Different feature extraction options extract different properties of the image. Some of the most common feature extraction methods include:

- **Intensity channel**

The intensity channel (or else known as grayscale) captures the intensity of each pixel in the image (usually on a scale from 0 to 255). Intensity images have been well known in the area of computer vision for their efficiency in terms of speed (since computations take place on one channel) but also for their efficiency in detecting key points such as edges and corners.

- **RGB channels**
  RGB images are three channeled images where each channel represents the intensity of the given color (Red, Green, Blue). In contrast to intensity channels, RGB channels can provide more information regarding object colours and illumination. For this reason (even though it depends on the application) RGB images are the most dominant in applications today.

- **Depth**
  Depth images are those which describe the distance of each pixel from the camera. They are extremely useful when objects need to be differentiated and distinguished in images. Common ways of achieving this includes using a pair of cameras(stereo imaging), in which every pair of points can be triangulated in order to calculate its depth.

Typically pedestrian applications in the earlier days extracted intensity images, although due to the advancement in processing power in systems today RGB is most commonly used.

Representation Factors

In the area of pedestrian detection, there are certain factors that need to be taken into consideration when thinking about features that are being extracted. In general, pedestrians tend to be recognized by their body parts (e.g. head, legs, body). Some of the most important factors which contribute in this representation include:

- **Human Pose**
  Pedestrians can be captured in a various numbers of poses. By looking at Figure 2.2, it can be noted how the pedestrians are captured in three different poses: from behind, from the left and from behind with a slight angle. Different poses result to different representations and therefore it is of a significant matter to take into account these variations when detecting. Various pedestrian applications usually set their cameras in a specific angle or from a specific distance in order to re-assure that the features extracted are in a certain range of poses (for example, a top-down view [33]).

- **Human clothing and accesories**
  Clothing and accesories can highly affect the way the pedestrian of interest

is represented. For example, in Figure 2.3, it can be noticed how the second pedestrian from the left holds a coat in front of her, while the fourth pedestrian from the left is wearing a long dress. Both of these occasions can affect what is considered as a more typical representation of a pedestrian (e.g. the first pedestrian from the left). In the same manner accesories such as hats and purses can affect other areas of the body such as the head and the torso, changing the representation.

Figure 2.3: Pedestrians with varying clothing([31]).

2.1.3 Feature Representation

The extracted image channels mentioned above are usually ineficient in order to represent a pedestrian feature-wise. This is mainly due to the fact that the features are acquired on a low level and are mostly robust when representing colour. Usually pedestrians and generally objects are better represented with higher level features such as corners, edges, histograms. These give a better idea of the scale, position and shape of the pedestrian.

Hand-crafted Features

Hand-crafted features have shown their effectiveness in computer vision and are the basis of many applications. In the field of pedestrians, HOG (Histogram of oriented gradients ([4]) features are a typical choice for the implementation. Although Hand-crafted features are efficient, they also have weaknesses in various areas. For example, HOG features are inefficient when trying to compute local features and is not scale and translation invariant.

Convolutional Neural Networks

Convolutional Neural Networks can be very effective in learning features of different situations regarding a certain task. Even though they are slower than handcrafted features, there way of interpreting data is build upon human brain neurons which make them more efficient in recognizing and describing objects. The general consideration when using convolutional Neural Networks is the organization and the number of training samples in order to efficiently be able to solve the task in hand.

2.1.4 Pedestrian Detection

When the features are extracted, an evaluation phase needs to take place in order to identify if a pedestrian exists in the give image or not. This can be done in two contexts:

- **Global context**
  In a global aspect a pedestrian could be evaluated based on his existence in an image. A global context detection is useful when the existence of the pedestrian in the image is in doubt and needs to be evaluated. Furthermore it could also contribute in processing time with respect to local contexts which would not compute features and attempt to represent pedestrians in an image which they do not exist. A binary classification diagram can be seen at Figure 2.4.

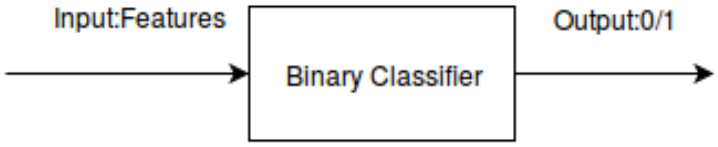

Figure 2.4: Global context diagram

- **Local context**
  In a local context the image could be evaluated pixel-wise in order to decide where the location of the pedestrian is in the given image. This step will provide the shape, size and existence of the pedestrian in order to further improve the estimation of the input. Unlike in a global context, the local context output would be an image of the same size as the input describing the estimation of the pedestrian location (Figure 2.5).

Figure 2.5: Local context diagram

2.1.5 Refined Result

The final stage of the pipeline deals with taking the estimation from the detection and refining the original input estimation. At this stage a good approximation of the pedestrian size and location should be made and therefore the output should be defined as the bounding box that encloses the area of the pixel-wise classification the best, although due to common factor in imaging (such as noise and distortion) the result may have to be re-evaluated to fit certain conditions. In pixel level classification this is commonly dealt by applying basic computer vision techniques such as erosion, dilation, opening and closing. The output should vary in size, features and confidence score with respect to the input.

Chapter 3

Related Work

The following chapter gives a brief overview of the work related to this project. It also gives a theoretical introduction to traditional and convolutional neural networks since they constitute the basic theory of this project. The chapter is split into two sections, one that analyzes related work based on hand-crafted features and one that analyzes related work on convolutional neural networks. Throughout Section 3.2 basic theoretical knowledge will be explained between related projects.

3.1 Hand-crafted features

Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters

A large ammount of previous work has been done on segmenting an image into regions, going back almost 40 years. [51] face segmentation as an edge-weighted linear graph. When seen in this manner, segmentation can be interpreted by the equation $G = (V, E)$, where each node $v_i \in V$ corresponds to an image pixel, and E denotes edges in images, which connect neighboring pixels. Each edge is assigned a weight based on a certain factor (e.g. difference between neighboring image intensities). A simple example of a graph can be seen at Figure 3.1.

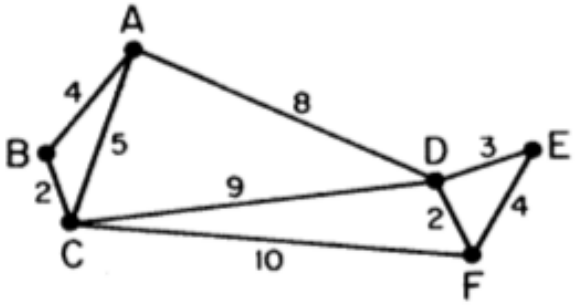

Figure 3.1: Graph containing six nodes and nine edges

From this interpretation a few other definitions are made in order to reach to the proposed method of segmenting an image into region, the minimum spanning tree (MST). In a edge-weighted linear graph, a path is defined as a number of edges that connect to a node (e.g. in Figure 3.1 sequence BADF or ABCFE). A circuit is a series of nodes which form a closed path (ABCA). A connected graph can be interpreted as a series of nodes that forms a path between any pair of nodes. From this a tree can be constructed which is a connected graph with no circuits which leads to a spanning tree , which in a given connected graph is a tree which contains all the nodes of the connected graph. A spanning tree of Figure 3.1 can be seen at the upper image of Figure 3.2.
As it can be seen from the graph, each edge is assigned a weight. A minimum spanning tree is a tree which sum of weights is the minimum among the graph . Image segmentation in this work is done by breaking MST's by thresholding the weight parameter.

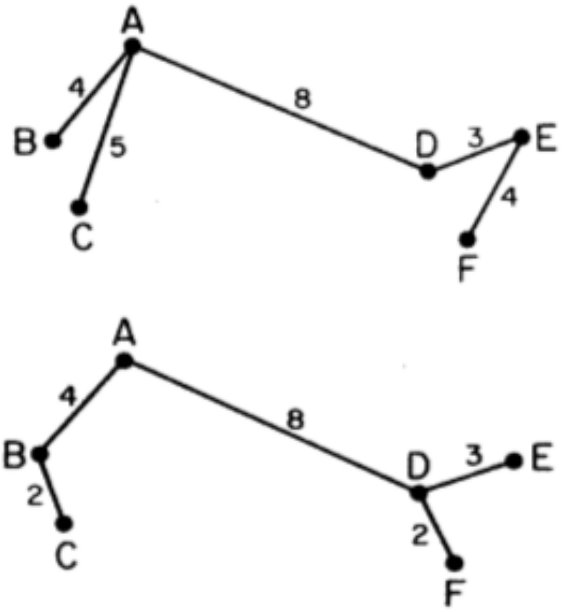

Figure 3.2: Upper image:Spanning tree of Figure 3.1. Lower image: Minimum spanning tree.

Efficient Graph-Based Image Segmentation

In [17], graph-based segmentation was used in a more efficient manner. Specifically, the proposed algorithm takes as input a graph $G = (V, E)$ with n vertices and m edges and outputs a segmentation $S = (C1, ..., C)$. The edges are sorted in a non-decreasing order. Afterwards an initial segmentation is made where each image pixel is assigned to its own component(region). Afterwards for each edge, its weight is checked in comparison to the biggest weight of each regions MST (denoted as BigW(1) and (2) in the pseudocode). if they are small, then those regions are connected. The pseudo code can be seen below.

```
//Sort edge weights into P = (o_{1} ... o_{m}), by non-decreasing order.
//Initialize segmentation S, where each vertex(image pixel) is in its
own component.
for i=1:m
w = P(i); //w contains two vertices of edge

        if(w - BigW(1) < thr && w - BigW(2) < thr)
                \\merge regions
        end
end
```

The proposed algorithm was tested both with a grid approach (assigning weights with respect to pixel differences on the edges) and a nearest neighbour approach (plotting pixels in space and assigning clusters). An example output can be seen at Figure 3.3.

Figure 3.3: Upper image:Graph output based on grid Lower image: Graph output based on nearest neighbour approach.

PedCut: an iterative framework for pedestrian segmentation combining shape models and multiple data cues

In recent years, techniques other than graphs have been used in order to segment regions of interest. One of those include the work in [16], in which pedestrians are attempted to be segmented from footage captured from mounted vehicles. This is achieved in a EM-framework, where in the E step the pedestrian is segmented by using unary and pairwise terms (by conditional random fields) and the M step constructs an Active Shape Model (Figure 3.4).

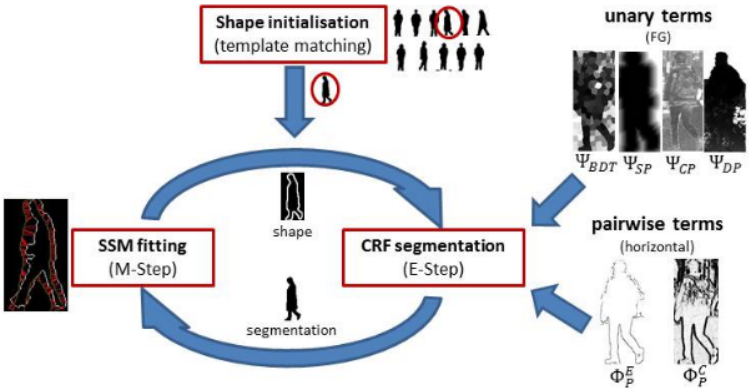

Figure 3.4: System diagram for pedcut segmentation.

The shape of the pedestrian is calculated as the first step by manually labeling 10946 pedestrian shape exemplars. Template matching is then performed in order to match the shape of the input. When that is computed, it is converted to its respective Multi Statistical Shape Model which is used as input to the conditional random field algorithm.

The disparity image for the input is calculated by applying a Semi Global Matching ([23]), along with the superpixel feature vectors. From this point four unary potentials are calculated. A sigmoid converted boosted decision tree trained on dense sift and Texton features, a shape potential which represents the distance transformation of the shape contour, a GMM-based color potential and GMM-based disparity potential which represents the median disparity over the segmentation calculated so far.
Along with the unary potentials, two pairwise potentials are defined. They include a color-sensitive potential, which increases the costs of an edge inversely proportional to the color difference of two pixels, and a contour-sensitive potential which increases the cost similarly but with respect to the edge magnitude of the two pixels (weight bases on disparity).
Once these are computed, the following energy function is minimized:

$$\begin{aligned} E(x, \Omega, I, D, S, \omega) = \sum_{i \in V} \omega_{BDT} \Psi_{BDT}(x_i, S) + \omega_{SP} \Psi_{SP}(x_i, \Omega) + \omega_{CP} \Psi_{CP}(x_i, I) \\ + \omega_{DP} \Psi_{DP}(x_i, D) + \sum i, j \in \epsilon \omega_P^C \Psi_P^C(x_i, x_j, I) + \omega_P^E \Psi_P^E(x_i, x_j, I.D) \end{aligned} \tag{3.1}$$

Where ω and Ψ denote weight of unary and the respective unary itself (e.g. BDT - Boosted Decision Tree, SP - Shape Potential, CP - Color Potential), I denotes color value, D disparity value, x and y pixels.

The M step consists of fiting the statistical shaped model to the obtained segmentation through an Active Shape Model ([3]). Chamfer matching is performed for calculating point correspondences between the image and the statistical shaped model.

3.2 Convolutional Neural Networks

This section describes the related work and theory needed to know behind traditional neural networks and convolutional neural networks for images. All references in the theoretical literature regardirg (both traditional and convolutional) neural networks are from [26] unless stated otherwise.

Multi-Digit Recognition Using A Space Displacement Neural Network

In [35], segmentation for digit recognition was attempted to be made over features maps obtained by a neural network rather than raw input image pixels.

Neural networks consist of layers of neurons connected to each other in a certain architecture. A neural network attempts to mimic the behaviour of the human brain, where each neuron receives inputs from its denditries and produces an output along its axon, which then branches out and is connected to other neurons. An example image of a mathematical representation of a neuron can be seen on the left at Figure 3.8, where the input (denditry) is a set of multiplications of $x_n w_n$ which are summed in a cell body along with a bias and following an activation function produce an output for the next neuron. If the output meets a certain threshold it can fire , meaning that it will send a signal to the next input that this part of the network is of interest.

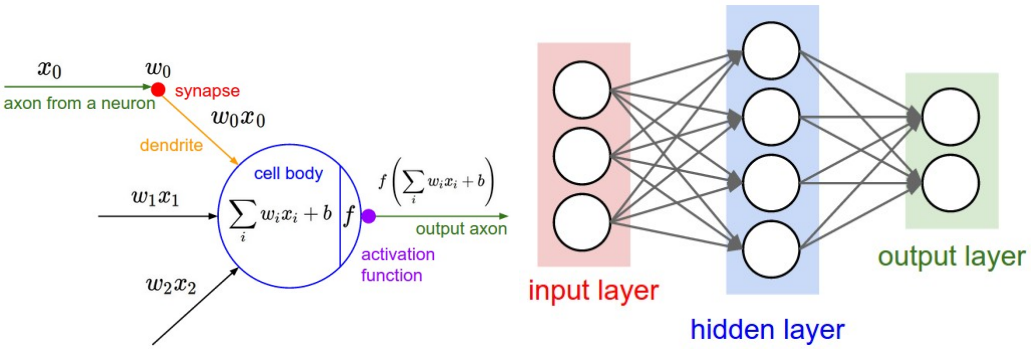

Figure 3.5: Mathematical example of a neuron (left) and a example of a neural network (right).

In the left part of Figure 3.8, x_n denotes the input, w_n denotes a weight factor and b a bias. The idea in neural networks is that through time the weights w_n can be learned and by feeding the right training data the network can adjust these weights in order to solve the task in hand through back-propagation of the outputs (chain rule). The activation function (also known as non-linearity) is a function which takes each input and performs a mathematical function upon it. Through time many activation functions have been used for neural networks such as sigmoid, tanh and relu. In the sigmoid function for example, each input is squashed into a range between 0 and 1.

The most simple case of a neural network can be seen on the right part of Figure 3.8 , where an input layer is connected to a hidden layer which then produces an output layer. The output layer could denote the result of a problem such for classification or regression. Notice that all neuros in one layer are connected to

all neurons in the next one but not connected between themselves. These type of layers are referred to as fully connected and will also be seen in the following subsection.

The network is broken in three phases, the training, the validation and the testing phase. The training phase takes part before the other two, where the network is provided with a series of data which are labeled in a certain way (for example in a network build for classification they would be labeled with an ID that represents a specific class). The output then is compared with the label and the weights of the neurons are adjusted accordingly (more details below on how that is done). The validation phase happens in the training phase (after a number of inputs have been fed into the network) and evaluates the accuracy of the network at the current time. Similarly the validation phase consists of sets of data and labels. Finally, the testing phase happens when training has ended and is when an input of unknown label is fed into the network and it attempts to make a prediction according to the data it has been fed during training.

The main factor that contributes in learning is the backpropagation, which in each forward pass (data passed through the layers) updates the weights in order to further adjust to the goal. The amount of adjustment is denoted by the loss function (more information in the next subsection). Back propagation is performed by using the chain rule, which is defined as backtracing a number of mathematical operations by applying their partial derivatives with respect to a specific variable . The theory behind backpropagation is of a large extent and therefore it will be only briefly mentioned in this project (a more analytical description can be found at [26]). For example, take the following equation:

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}} \tag{3.2}$$

Which describes the sigmoid activation function for a two-dimensional neuron. An example of this function can be seen at Figure 3.6. The forward procedure is the combination of the mathematical operations need to be made in order to compute $f(w,x)$ (numbers in green), which result is squashed in a range between zero and one . The backward propagation (denoted in red) at each step is the partial derivative of the input with respect to the variable of interest times the current propagation result. For example, in the exponential step, the result is equal to $e^{(-1)} * -0.53 = -0.20$.

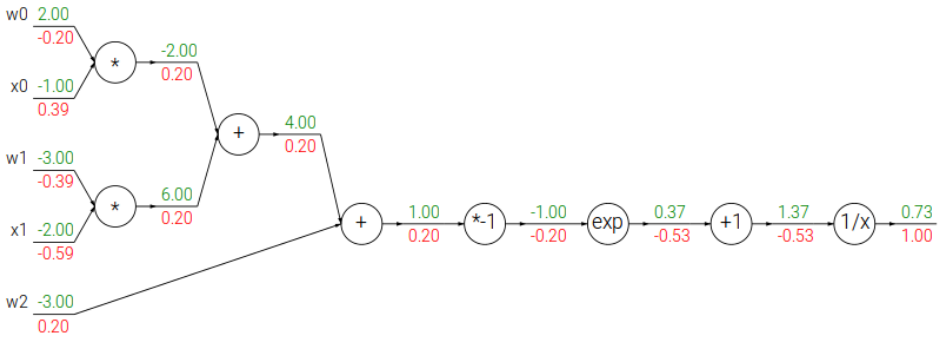

Figure 3.6: Example diagram of a 2D neuron with a sigmoid activation function.

Neural networks have also been extended in the area of image processing. The neural network in [35] consists of four convolutional layers, which convolute an input image with a certain filter (weights - more information regarding convolutional layers can be found in the next subsection).

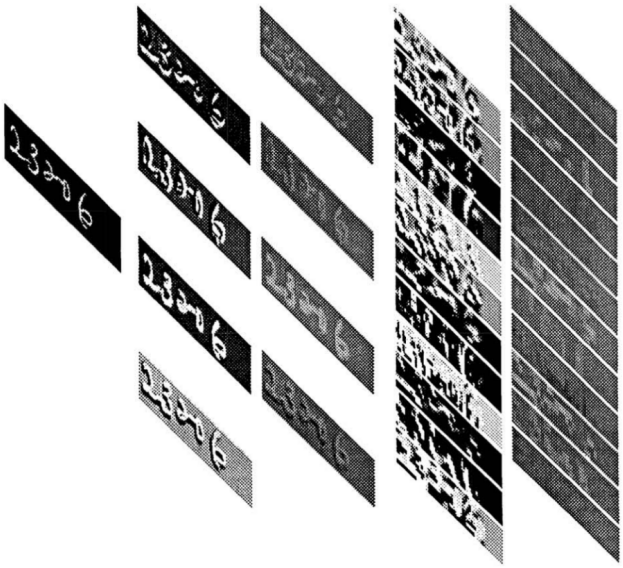

Figure 3.7: An input image of digits and its four convolutional outputs.

The network is also able to handle multiple digits as input (as long as the number is specified by the user). This is dealt with by increasing the width of the last feature map with respect to the number of digits. Since hand-written digits width can vary, the authors propose alternate output vectors of varying feature map widths (2,3,4 and 5 which is the default).
The output of this network is then coupled to a Viterbi alignment module which is responsible for choosing the best representation of the input.

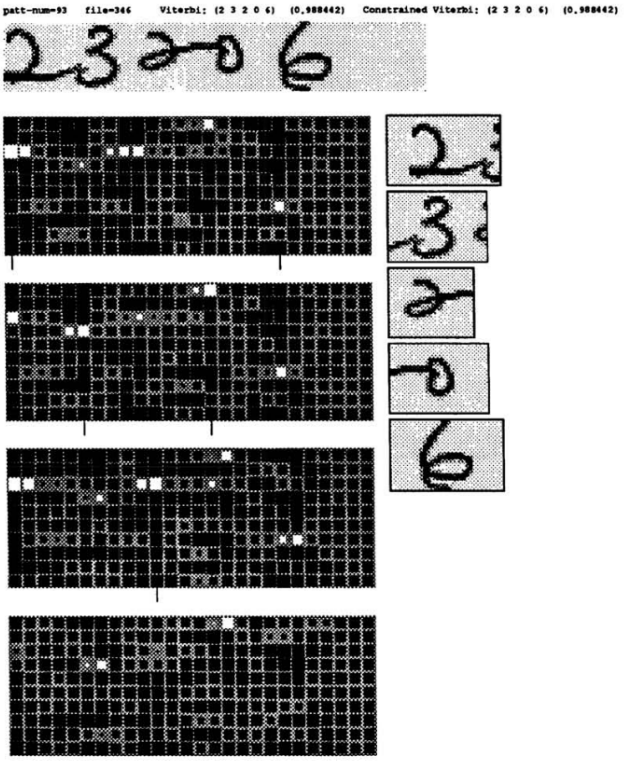

Figure 3.8: An input image of digits, its four width outputs(5,4,3,2 from top to bottom) and the cropped recognized output.

Simultaneous Detection and Segmentation

The work in [22] proposed a combinational detection and segmentation framework by combining convolutional neural networks and a SVM classifier.

Convolutional neural networks are an extension of neural networks adapted for applications related to images. In this context, the idea behind convolutional neural networks has drifted from the biological brain and has adapted more to solving machine learning tasks. Due to how images are represented, every neuron in this case is a three dimensional input, therefore the output and the layers is also mapped in three dimensions (height, width, depth - unlike the right part of Figure 3.8 which is one dimensional). The division of dimensions control the number of weights for each layer and prevents them from reaching overwhelming numbers.

Network Layers

A convolutional neural network can be defined in a various number of ways and its architecture (structure of layers) depends on the task. Even though networks differ from each other, there are a few types of layers that are most commonly seen in every convolutional neural network. Those include the following.

- **Input layer**
  The input layer consists of an image, which is represented by the three dimensions mentioned above and therefore would be of size WxHxD. Typically the input is pre-processed before entering the network for efficiency reasons. The most common methods regarding pre-processing include the following.
  - **Image Resizing**
    In order for the weights to fit the image correctly (explained more below), all images in all phases are re-sized to a certain size. These images are usually resized using methods such as bilinear or bicubib interpolation.
  - **Mean subtraction**
    The most common form of pre-processing in convolutional neural networks is mean subtraction, where an image exists which represents the mean value of the whole training dataset. This image is then subtracted pixel-wise to each image that goes into the network. Mean subtraction achieves centering the cluster of the data in each dimension to the origin.
  - **Normalization**
    In normalization the data dimensions are normalized so that they fit a certain scale. This is most typically done by dividing each dimension by its standard deviation after mean subtraction has taken place.

  Other (not so common) forms of pre-processing include PCA (dimensionality reduction) and whitening (another form of scale normalization).

- **Convolutional Layer**
  Convolutional layers form the core operation for the network. In the simplest way explained, a set of filters (which are the learnable weights of this layer) are convolved with the input in order to produce the output of the layer. Throughtout time the filters will activate when they see certain features (depending on the labeled training data). A convolutional layer consists of four main factors:
  - **Number of filters**
    The number of filters defines the depth of the output. Each filter is extended along the depth of the input (for example the first convolutional

layer would have filters of depth 3). Filters are relatively small in terms of spatial size (depending on the input size) and are convolved across the image in order to produce the result.

- **Spatial Extent**
  The spatial extent corresponds to a factor which denotes the size of the filter. This is usually set with respect to the input. Smaller kernels usually perform better with smaller images and vice verca.
- **Stride**
  Stride denotes how close to each other in terms of coordinates are the filters. A stride of 1 for example denotes that each filter will be set 1 spatial unit apart, resulting to a very large output and many overlapping regions.
- **Zero padding**
  When filters do not fit across the input the borders are usually padded with zeroes in order to assure spatial consistency.

The output of a convolutional layer in terms of size then is denoted as:

$$\begin{aligned} Width &= (W - F + 2P)/S + 1 \\ Height &= (H - F + 2P)/S + 1 \\ Depth &= K \end{aligned} \tag{3.3}$$

Where W and H denote width and height of the input respectively, F the spatial extent, S the stride and P zero padding. An example of can be seen at Figure 3.12.

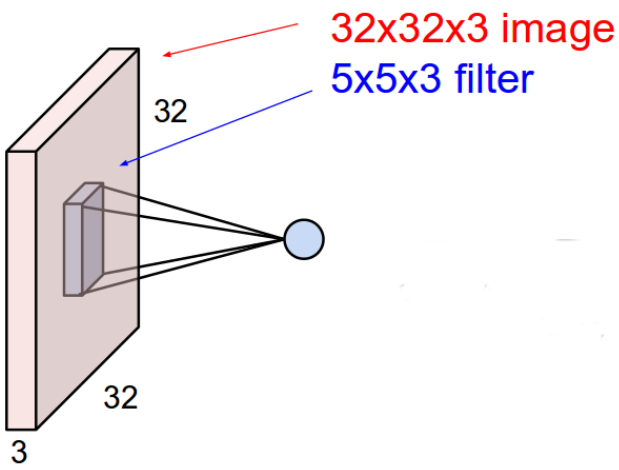

Figure 3.9: A 5x5 filter positioned in the center of a 32x32 image.

- **Activation Layer**
  The activation layer has been talked briefly in the previous subsection of neural networks. There exist many activation functions but in this project only the ReLu non-linearity is used and therefore it is the one that will be briefly discussed.

  The rectified linear unit (ReLu) computes the function $f(x) = max(0, x)$. Its input is the output of the previous layer (usually the convolutional layer) and thresholds the activation at zero.

- **Pooling Layer**
  When dealing with images and convolutions it is almost certain that large amount of features will be resulted from the convolutional layers which can cause overfitting and large data. Therefore the most common way to downsample these outputs is to insert a pooling layer after each convolutional layer. Each depth slice of the input is resized spatially using a certain pooling method. The most common methods include average, L2-norm and max pooling. In this project and in most cases max pooling is the dominant choice in convolutional neural networks.

  Just as in convolutional layers, a pooling layers requires a filter size and a stride size. The idea behind the stride option is the same as in the convolutional layer. In the filter size, if for example, a 2x2 size is selected, the filter will downsample every 2x2 region of the input and select the dominant value out of these 4 pixels. The output in height and width is computed just as in the convolutional layer (without adding zero padding), although the depth does not depend on the filter size, but remains unchanged with respect to the input. The output dimension equations and an example of max pooling can be seen below.

$$\begin{aligned} Width &= (W - F)/S + 1 \\ Height &= (H - F)/S + 1 \\ Depth &= D \end{aligned} \tag{3.4}$$

  Where W, H and D denote width, height and depth of the input respectively, F the spatial extent, and S the stride.

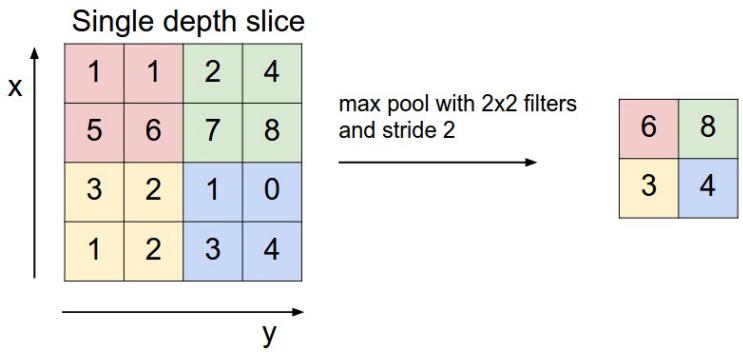

Figure 3.10: Example of max pooling with a 2x2 filter and a stride of 2.

- **Fully Connected (Inner Product) Layer**

  Similar to the simple network shown to the right at Figure 3.8, a fully connected layer is a layer that has full connections to all the outputs of the previous layers. Its activations are computed as shown in the left side of Figure 3.8 (a matrix multiplication and an addition of a bias offset). Fully connected layers are most commonly used in order to classify the input image into a certain class (e.g. a classification problem of 1000 classes would end in a fully connected layer of 1000 outputs) or reduce the image representation into a one dimensional vector. Fully connected layers are very similar to convolutional layers, and each fully connected layer can be transformed into a convolutional one and vice versa.

- **Loss Layer**

  Throughout training, the weights of each layer need to be adjusted accordingly in order for the network to learn the task in hand. The layer that defines how close the output of the network is in comparison to the label provided is the loss layer. It produces an output at each pass which is used in back propagation in order to adjust the weights properly. Most common loss function include softmax, SVM and euclidean loss. In this project the softmax loss has been chosen.

  The softmax loss is computed in two steps:First, the softmax is computed which essentialy takes as input the scores of the last layer, computes the exponential and normalizes it so the sum of each score adds up to one (so each class's score is described in a probabilistic manner). The softmax equation is shown below:

$$f_j(z) = \frac{e^{zj}}{\sum_k e^{zk}} \tag{3.5}$$

Where j denotes the j-th element of scores in z, and k denotes the number of classes.

This is then used in order to compute the loss for the specific class of interest (denoted by the labels). The goal is to maximize the probability of the desired class through time. Therefore by computing the cross-entropy loss, the network learns how far away from the goal it is. The cross-entropy loss can be interpreted as below:

$$L_i = -log(f_j(z)) \tag{3.6}$$

A simple example of a softmax loss computation can be seen at Figure 3.11:

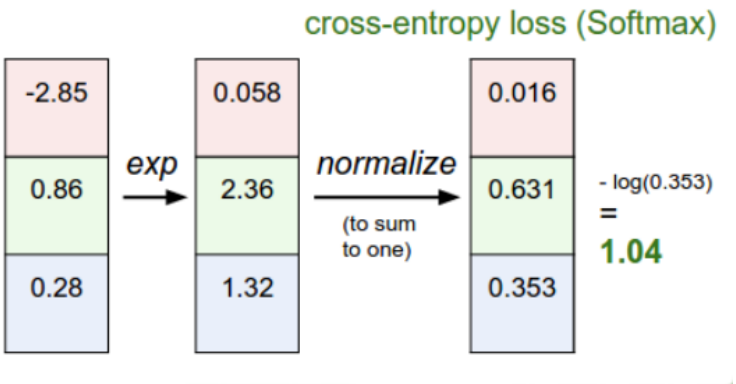

Figure 3.11: Example of softmax loss computation for a 3x1 vector.

The system diagram of [22] can be seen at Figure:

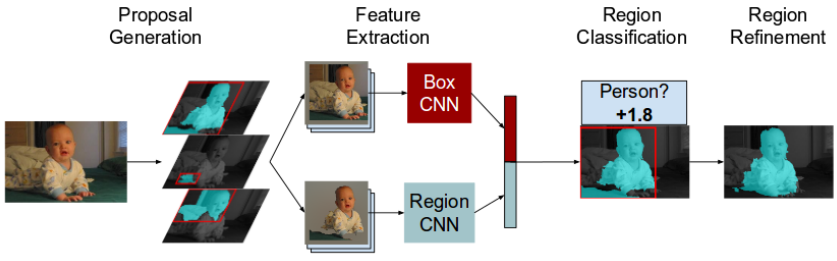

Figure 3.12: Detection and segmentation diagram

The first step of the pipeline involves generating region proposals of the input image. For each image 2000 region candidates are generated by using the method in [2]. Specifically region proposals are computed in multiple resolutions which are then fused into a single multiscale hierarchy at the finest scale. Regions are then combined from both the single scale and the multiscale hierarchy.

For extracting features, initially two separate convolutional neural networks are trained, one for bounding box detection and a second one for segmentation (through the proposals created explained above). The networks architecture is similar with the one in [27], which consists of five convolutional layers (followed by activation and pooling layers), three fully connected layers and a softmax loss layer. Each network is fine-tuned from the networks trained from this work, and afterwards a single network architecture is constructed (Figure 3.13).

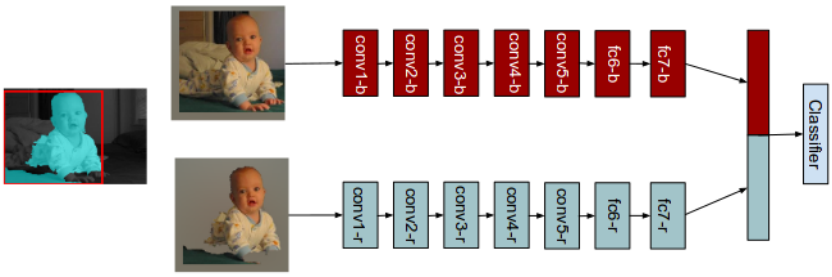

Figure 3.13: Detection and segmentation diagram

The last classification layer is removed after training and features are extracted from each network from the seventh layer (fully connected layer) which are then concatenated and are inputed to a linear SVM classifier.

The SVM classifier is trained by using ground truth as positives and region proposals overlapping ground truth by less than 20% as negative. The authors then re-estimate the positive set by using the highest scoring region proposal for each ground truth that scores more than 50%. The classifier is re-trained by using this set. This method showed major improvement in accuracy in comparison to just using the first method.

In the last step, the region is refined for higher precision. The bounding box of each predicted region is taken, padded and then splitted into a 10x10 grid. A linear regression classifier is trained for each grid cell in order to compute the probability if this cell belongs to foreground or background.

Fully Convolutional Neural Networks for Semantic Segmentation

The work in [32] proposes a convolutional neural network for semantic segmentation by training a network end-to-end, meaning that the network itself is responsible for the final segmentation output, unlike the work described above.

Prediction is done on a pixel-wise level instead of a image-level through converting fully-connected layers and adding de-convolutional layers. When transforming fully connected layers to convolutional, a heat map can be outputed as an end result before the loss function (Figure 3.14).

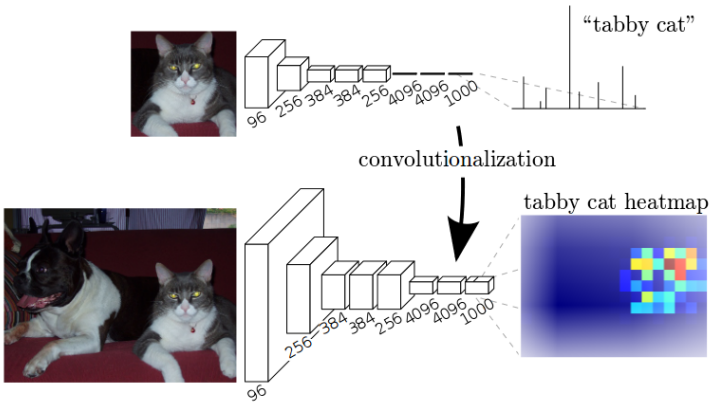

Figure 3.14: Fully convolutional network fully connected layer conversion

Deconvolutional Layer

As a convolutional neural network goes into deeper layers, spatial size decreases. A deconvolutional layer is commonly used in order to upsample layer outputs to a specific size for segmentation. This is useful for example when the goal is to train a network that performs a pixel-wise classification and the label is an image instead of a class. Essentialy a deconvolutional network performs the same operation as a convolutional network but backwards, meaning that in the forward pass of the network the deconvolutional layer will perform the same operation as a convolutional layer at its backward pass and vice versa.

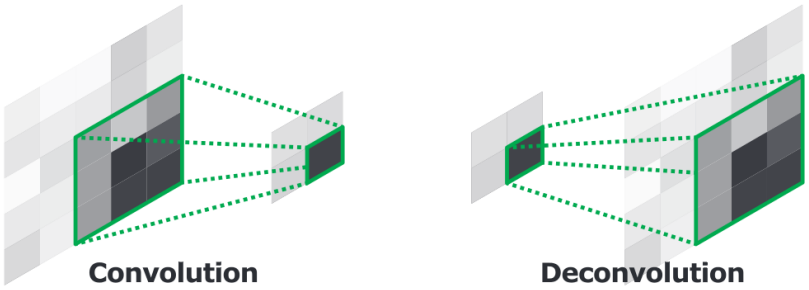

Figure 3.15: Convolution and Deconvolution [36]

An example of how an image is re-constructed to a larger resolution and its feature maps can be seen at Figure 3.16.

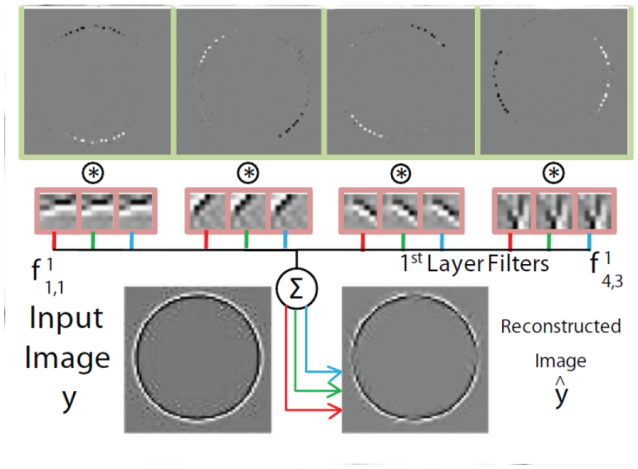

Figure 3.16: Deconvolution layer for a given input image ([53]).

As it can be observed, different feature maps activate at different areas.

The authors in [32] propose three different networks (Figure 3.17) based on the architecture of [41]: One that follows the structure in [41] until the seventh convolutional (or third fully connected converted) layer and then inserts a deconvolutional layer in order to upsample the heatmaps and predict the final result, one that uses the final layer and the fourth pooling layer in combination and one that uses the final, the fourth and third pooling layer for segmentation prediction.

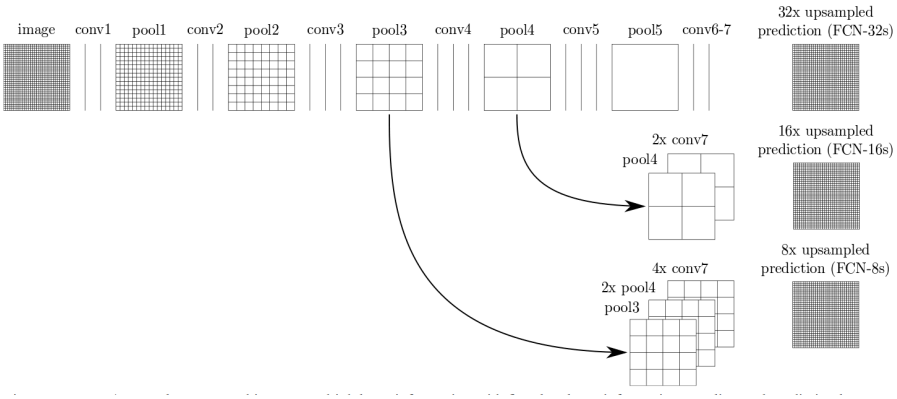

Figure 3.17: Fully convolutional network approaches

As Figure 3.18 shows, the first network (32s) provides more semantic information where the other two (16s and 8s) provide more fine detais. The results show that the layers that combined more features for the final prediction ahieved higher accuracy.

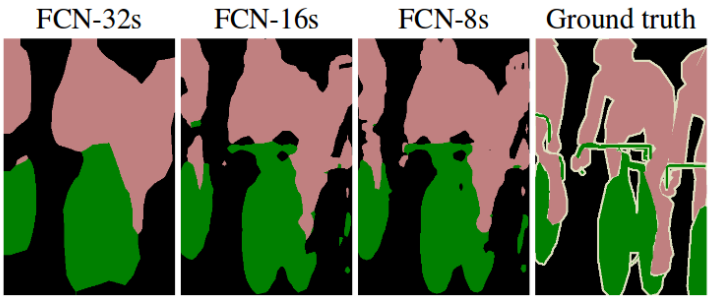

Figure 3.18: Fully convolutional network approaches results

Chapter 4

Final Problem Statement

The following chapter will begin with declaring the problem statement of the project. Afterwards a number of limitations will be outlined followed by the systems requirements. Last, the system design will be outlied and the datasets used will be mentioned.

4.1 Problem statement

Given the introduction and initial problem statement, the following problem statement can be made:

How can a system be designed which is able to detect the location of pedestrians given a bounding box coordinate and refine the results?

4.2 Limitations

The projects limitations have to do with the factors that the system is not able to handle and errors that are naturally expected to occur.

Non-known enviroment/condition Due to the robustness that is desired to be obtained and the need to combine datasets (due to the lack of training data) the conditions and enviroment that the images are captured during testing are not known, meaning the input can be taken from any street or any place, therefore the system will not make any attempt to adapt to them during test time. Also various cameras are used and therefore varying resolutions and camera settings are expected.

Pedestrian Segmentation In an ideal scenario the segmented pedestrian in test time would be a representation of the pedestrians location and size, although it is

expected due to external factor this to not be the case. Therefore some parts of the resulted segmentation will be discarded.

Number of Pedestrian Due to the fact that only a single pedestrian is segmented during training, the same assumption will be made during testing and only one pedestrian will be attempted to be detected and re-evaluated.

Re-adjustment error The system is not expected to work in all cases, meaning that in some scenarios the re-adjustment might not be precise (mostly due to the challenging scenarios and representation factors). Therefore a margin in the re-adjustment should be accepted.

4.3 Requirements

The requirements of this project are the key points which will result to a correct behavior and not cause the system to output wrong results.

Pedestrian Existence Even though the system is trained on negative examples, it is considered a requirement for the system that the input image should include a pedestrian, or else results may vary significantely.

Number of Pedestrian Due to the limitation mentioned above, it is required that only one pedestrian will be recovered.

Time of execution The time of execution for the refined result should be of an acceptable rate.

Precision Score and Confidence Score The system is expected (without taking the error margin mentioned in the limitations) to perform at least to the same resulting scores as the inputing algorithm provides.

4.4 System Design

The main outline of the system design can be shown in Figure 4.1. The system is split in two categories, an offline one in which the network for pedestrian segmentation is trained and an online one where a bounding box image is given in real time and is attempted to be evaluated/refined.

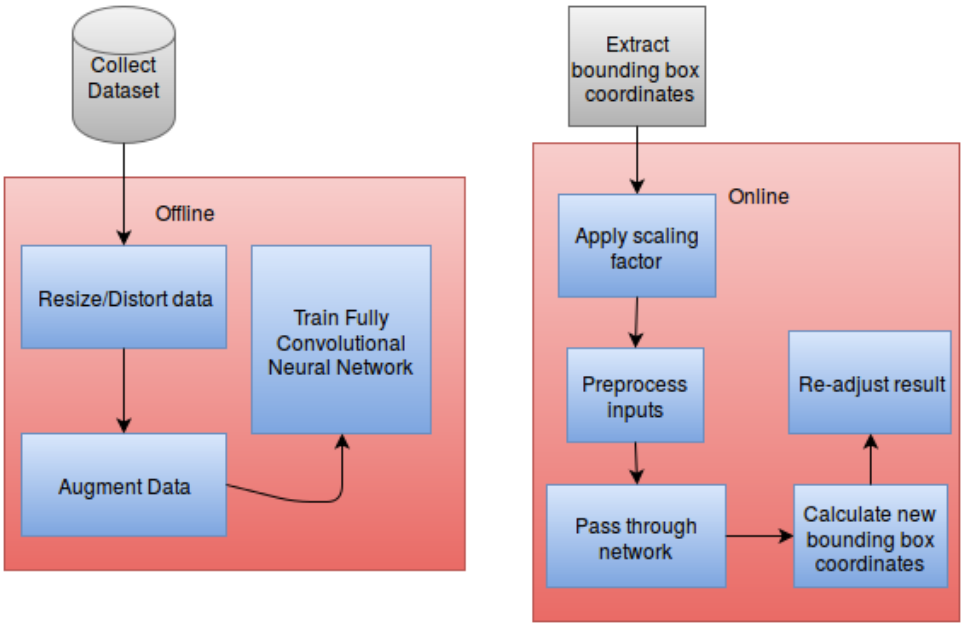

Figure 4.1: System Design

4.4.1 Offline

Collect Dataset

A collection of datasets are collected before-hand in order to train the network efficiently. These include positive examples (datasets of pedestrians images and their segmented labels) and negative examples (datasets of non pedestrian images and their segmented labels).

Resize/Distort data

Different pedestrian images can have different resolutions, therefore all training data should be fit to a certain size and fixed to a certain resolution.

Augment Data

In order to increase the training samples, the data is augmented in a number of ways without overfitting the network.

Train Fully Convolutional Neural Network

The fully convolutional neural network is then trained with the training data collected and with certain properties.

4.4.2 Online

Extract Bounding box

The bounding box of the original prediction is given in the beginning by the corresponding detector.

Apply Scaling Factor

The bounding box coordinates from the detector are likely to not be ideal and therefore in order for the network to clearly refine the result a scaling factor is applied to the bounding box

Preprocess input

The input is then preprocessed just as the training data is in order to get better results.

Pass through network

The input is then passed through the network and evaluated on a pixel level.

Calculate new bounding box coordinates

The result from the network is used in order to calculate the refined result.

Re-adjust result

The bounding box is re-adjusted based on the output of the network.

4.5 Datasets

Offline

A number of datasets are combined in order for the network to learn efficiently to segment pedestrians on a pixel-level. Datasets are divided between fully visible and partially occluded pedestrians.

It is worth noting that the ratio between the data is 65% fully visible pedestrians and 35% partially occluded pedestrians. In [11], it is noted that 53% of the pedestrians detected are ocluded in some frame, with 29% never being occluded, making therefore an occluded dataset necessary for good results. Additionaly, the ratio between training and validation data is 80% to 20% respectively. This is due to the fact that the networks main objective is to learn the data but it is also significant to validate in order to understand how well it is performing while training.

The datasets used that include pedestrians include three and are mentioned below.

Penn-Fudan dataset

The Penn-Fudan Database for Pedestrian Detection and Segmentation ([44]) consists of 170 images of 424 labeled pedestrians captured around the university of pennsylvania and the fudan university from a stationary camera. It includes only walking pedestrians which are fully visible. Each image is of medium to high resolution (average from 300x360 to 600x600 pixels) and is labeled in a map of the same size as the input image with indexes from 1-K , where K is the number of pedestrians that exist in the image (this number varies from image to image). An example can be shown at Figure 4.2, where the three pedestrians are labeled appropriately pixel-wise in the segmented image.

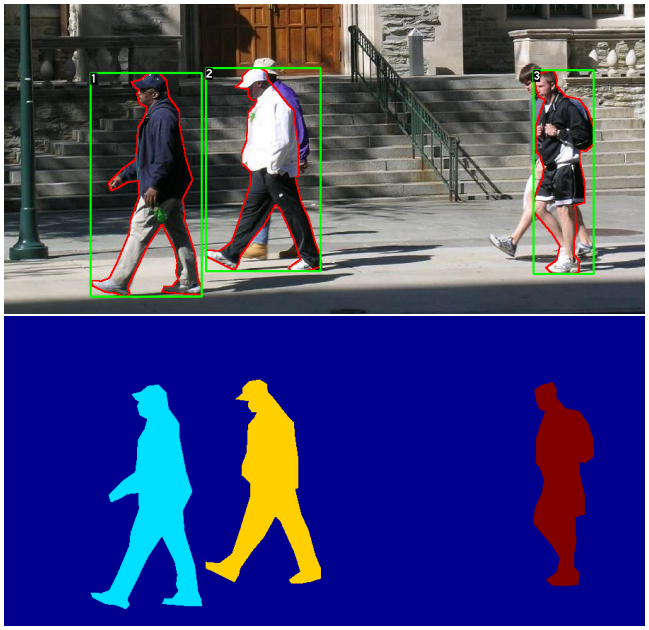

Figure 4.2: PennFudan dataset example - image and label

Daimler Pedestrian Segmentation Dataset

The Daimler dataset ([16]) consists of 500 enclosed pedestrian images captured from a vehicle-mounted calibrated stereo camera rig in an urban environment. For each image a PNG image, a disparity and a ground truth pixel-wise labeling is provided, although the disparity map in this project is not used. Each label is indexed either between 0 (background) or 255 (foreground - pedestrian). Each image has a height between 34 and 468 pixels and a width between 11 and 267 pixels. Ad-

ditionaly, each pedestrian image is given 10 pixels of background padding away from the pedestrian location.

Figure 4.3: Daimler dataset example - image and label

Custom partial-occluded dataset

To the authors knowledge, there does not exist publicly available a partial occluded pedestrian segmentation dataset. Occlusion is a major issue in pedestrian detection and there does not exist a straighforward way of dealing with it ([11]). Therefore, a custom database is necessary to be made. The dataset in [24] is used since they provided roughly 500 partially occluded bounding boxes of partialy occluded pedestrians. 212 are manually labeled and used in order to deal with occlusion.

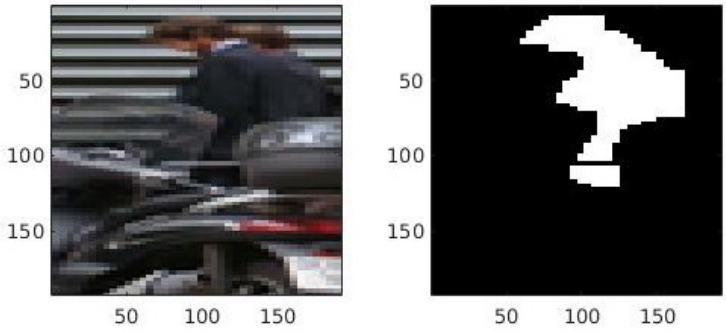

Figure 4.4: Custom partial occluded pedestrian dataset example

Icome dataset

The icome dataset ([50]) consists of a number of images of people in various clothing and positions and there respective segmented labels. Even though the data is not exclusively pedestrians, the network is pre-trained on this dataset and learns the network how an overall human shape should look like.

Figure 4.5: Icome dataset example - person image and label

Online

Four datasets widely used in evaluating state-of-the-art pedestrian detections [11] (Caltech [7], INRIA [4], ETH [12] [13],Tud-Brussels [**tud_bruss**]) are used in order to evaluate the networks performance. Each of these are composed of videos taken in streets of big cities (e.g. California, Brussels) where the location and size of pedestrians are manually labeled as ground truth labels. In some benchmarks occlusion,difficult to recognize pedestrians and large clustered groupped pedestrians are also denoted (Caltech). Some screenshots from the benchmarks can be seen below.

Figure 4.6: Image samples from Caltech(USA - Top Left), ETH (Top Right), INRIA (Lower Left) and Tud-Brussels (Lower Right).

Chapter 5

Design

The following chapter focuses on the explaination of the projects design. The chapter is divided in two sections, the offline which entails details regarding the construction of a fully convolution neural network and the online one which explains the procedure during testing.

5.1 Offline

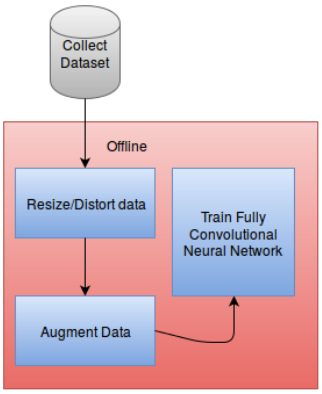

Figure 5.1: System design (offline module)

In order to refine detections a series of decisions need to be made. This involves the selection of training data, the representation of this data and the network architecture/parameters.

5.1.1 Architecture

The networks architecture can be seen at Figure5.2:

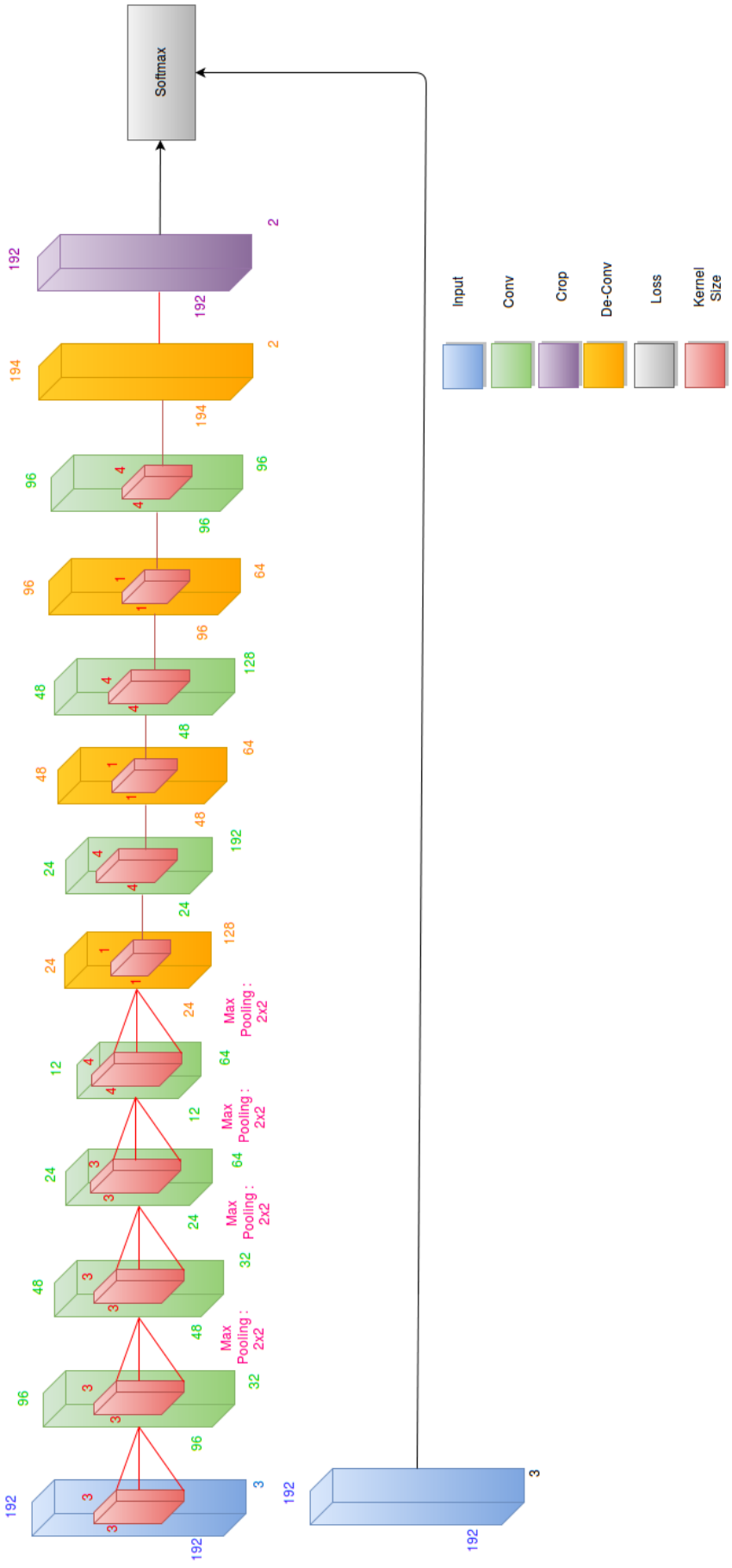

Figure 5.2: The proposed network architecture.

It is worth noting that the upper input denotes the data (the image itself) and the bottom layer denotes the image label.

The network has to be deep to some extent. This makes it possible to decrypt different type of information. Earlier layers tend to provide low level features such as corner and edges, where deeper layers provide more semantic information such as shapes and object contours ([52]). At the same time though, going deeper also means that feature maps become smaller in size, so a very small feature map (e.g. 1x1,3x3) would not provide valuable information. For this reason, the smallest size of the network is a 12x12 map.

The network is comprised of eight convolutional layers, four de-convolutional layers, one crop layer and a softmax loss layer.
The first five convolutional layers are set with a filter size of 3x3. Small filters are useful and have proven their efficiency for convolutional neural networks, since they decrease the number of parameters the output layer will have while also effectively extracting features at the same time. Furthermore, in every of the first five convolutional layers features are convolved three times, where after each convolution a rectified linear unit (ReLu) function is applied. Applying more convolutions with smaller filters instead of a single convolution with a larger filter has proven its efficiency in the past due to the decision function playing a bigger role and decreasing parameters ([41]). After each convolutional layer a pooling layer is added for downsampling. Note that the fifth convolutional layer does not have a max pooling layer due to its input dimensions being already small in spatial size (12x12), therefore in the figure of the architecture it is not included (meaning that the fifth convolutional layer in the image is actually the sixth one in the network). Each convolutional layer and all de-convolutional layers excluding the last one are padded with zeros in a 1x1 dimension around the image. The strides in the pooling and deconvolutional layers are set to two. This also minimizes parameters and allows a deep network without overfitting.

Each deconvolutional layer is an re-constructed upsampled version of its previous convolutional layer. After the upsampling takes place, the layer is concatenated with the convolutional layer that has the same spatial size (e.g. the second deconv layer is concatenated with the second conv layer, the third deconv layer with the first - before the pooling layer). The reasoning behind this choice is due to the fact that when dealing with low resolution images and segmentation methods it is important to combine both early and late layer information. This method is often seen in super-resolution for converting images from low to high resolution. Furthermore, the concatenation deals with information that can be lost at the images

boundaries due to deep convolutions (small feature maps).
Following each de-convolutional layer a convolutional layer is placed that uses a filter size of one. This layer serves as a dimensionality reduction tool and periodically reduces the dimensions of the image in depth from 192 to 2 (which is the end goal for the classification). A figure which visualizes the concatenation procedure can be seen at Figure 5.3.

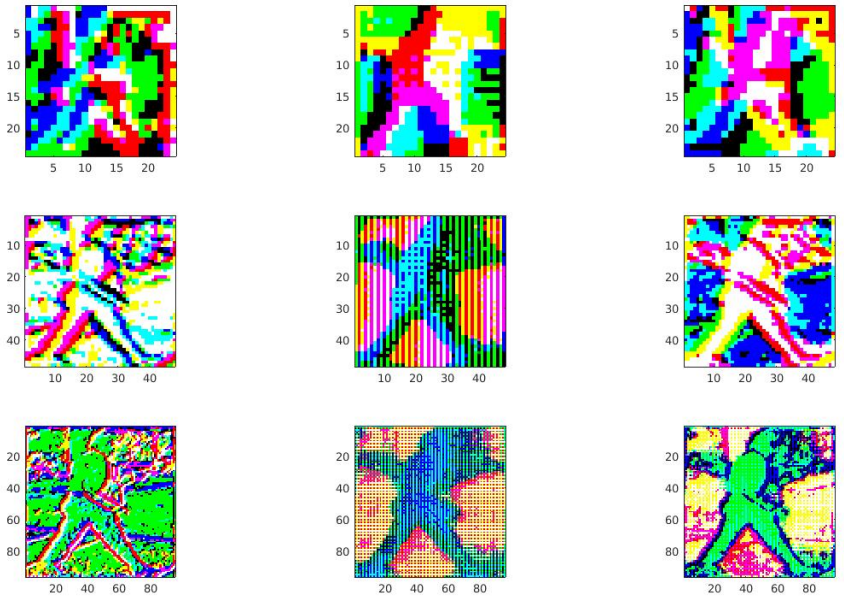

Figure 5.3: Upsampling/concatenation procedure. From top to bottom: fifth,sixth and seventh layer. From left to right: Convolutional layer, De-convolutional layer and concatenated layer.

Each layer from left to right contributes in its own way in order to efficiently segment the pedestrian. The convolutional layer, even though it contains a large ammount of noise, is able to detect corners and edges throughout the image. The de-convolutional layer manages to extract semantic information (the general shape of the pedestrian) by re-constructing its input layer, although still containing noise. Finally the concatenated layer combines both of these properties in order to visualize the area of the pedestrian.

Due to the upsampling in the last layer slightly exceeding the label size, a cropping layer is added. Note that the cropping is small in size (2x2) and thus should not affect performance. These are then fed into the softmax loss function in order to compute how close the networks weights are to solve the task.

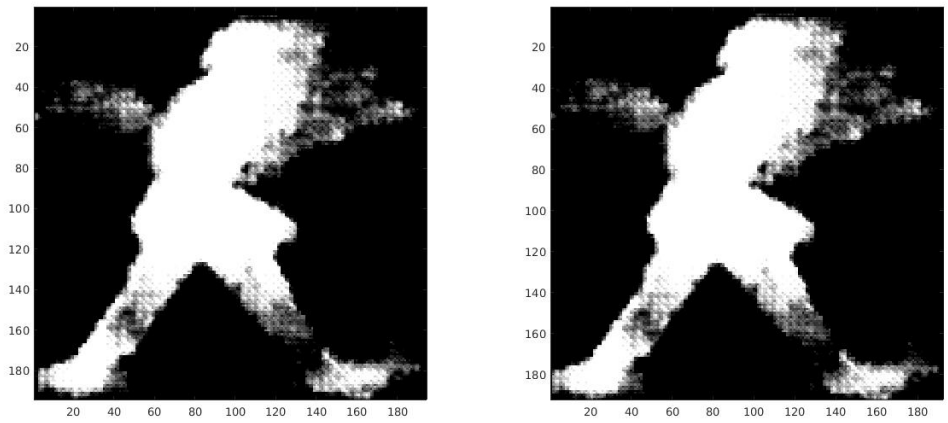

Figure 5.4: Left: Score layer (foreground channel). Right: Crop layer Output

5.1.2 Resize/Distort data

The datasets mentioned in Section 4.5 are modified to a certain extent. All training data should be labeled in a pixel-wise binary classification manner, meaning a pixel that has a value of zero is considered as background where a pixel that has a value of one is considered as foreground. This is significant in order for the network to learn for which features to activate in test time. For the PennFudanPed dataset, all pedestrians are cropped from each image and their corresponding mask images are converted to a binary label. The same idea is used in the partial occluded dataset. The daimler dataset already provides cropped bounding boxes and therefore no pre-processing is made.

The size of the input image to the network is another significant matter. As mentioned in Chapter 2, when dealing with pedestrian images taken from surveilance cameras the resolution of the input can vary widely. Also, the camera parameters are unknown. Therefore there should be a compromise of the input size in order to take into account low and high resolution images. The input size in this project has been chose to be 192x192.

Other than the size, there needs to be a decision making also regarding the res-

olution of the input. The datasets used contain images in a good resolution since it is important to capture details when constructing a dataset. The resolution cannot be expected to be of the same quality in test time though. Training a network strictly on images that tend to have a good resolution will result to false negatives in test time, therefore both the training and testing data is resized in two ways: First a bicubic interpolation takes place and each image is resized to a 40x80 size.

Bicubic interpolation is known as one of the most efficient method in image resizing, due to the fact that for every pixel, an attempt to reconstruct the exact same surface on the resized image is made. The input image is zero padded and translated according to the resized factor. Then for each pixel the weighted average of the two translated values on either side is computed in order to compute the new result.

Afterwards a nearest neighbor algorithm is used in order to upsample the bicubic resized image to the desired 192x192 size. Nearest neighbour's performance is significantly worse than the bicubic. It chooses the nearest neighbour after the translation instead of calculating a weighted average. A comparison of the bicubic, bilinear and nearest neighbour approach along with a low resolution image can be seen at Figure 5.5. The nearest neighbour approach imitates the noise of the low resolution image the best in comparison to the other methods.

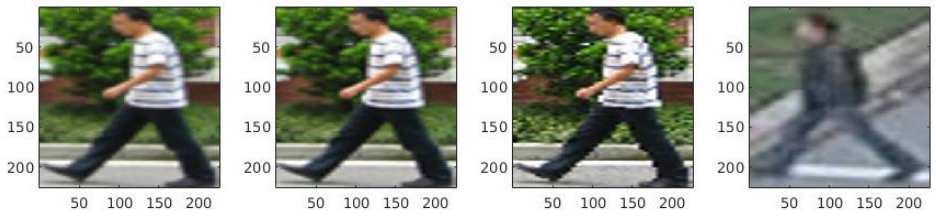

Figure 5.5: Example of resizing an image with different methods. From left to right: bilinear interpolation, bicubic, nearest and a low resolution image for comparison.

In convolutional neural networks it is highly recommended to use the same pre-processing methods for both training and test stages. The idea behind the two re-sizing methods makes sense also in test time. If the image is of high resolution, it needs to be re-adjusted in order to be similar to the training data. If the image is of low-resolution, it will be upsampled to a certain degree with the best resizing technique , and the nearest neighbour approach will not have such a large effect on it.

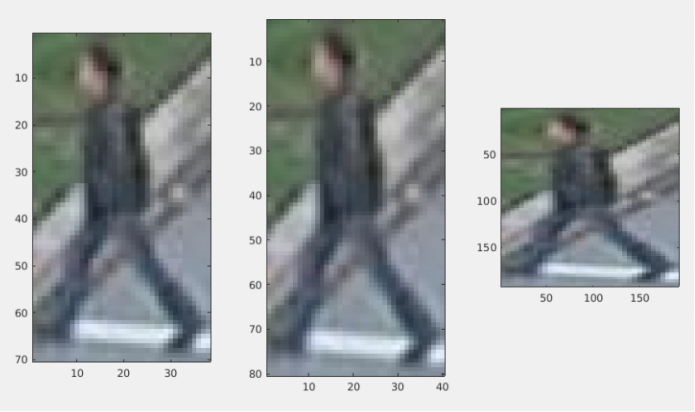

Figure 5.6: The two resizing methods performed on a low resolution image.

5.1.3 Augment Data

Data augmentation is one of the most popular methods in supervised learning and in convolutional neural networks in order to efficiently increase training samples for higher accuracy and at the same time avoid overfitting ([27],[41],[52]).
As noted in Section 4.5, the number of positive training images is relatively small. It is almost impossible to learn a network with such a small amount of data, so therefore data augmentation is also necessary in order to increase samples. Other than this, it is also significant which augmentation methods are chosen and how much weight is given to them (this depends on the task). For example in [5], the authors network achieves rotation invariance by rotation each training example by a random degree (0-360).
This is applicable in their situation since the training samples consist of images taken in space, thus rotation will give samples that are also realistically possible to occur in test time. In this projects case, for example rotating a pedestrian by 100 degrees will not result in a realistic test sample.

The data is augmented in a various number of ways, which include:

- **Scale**
  Scaling is likely the most important augmentation method for this task. Detection results tend to incorrectly estimate the size of a pedestrian due to the challenging factors mentioned in Chapter 1. Therefore providing a realistic number of scaled versions of each image would contribute in a more efficient network. The network scales range from a value of 1.6 to 1 and decrease in step by 0.18. Any higher number than 1.6 starts to be considered inefficient since the representation of the pedestrian becomes more and more distorted and begins to look more and more like background.

Figure 5.7: An example of a training sample scaled at six different factors (from 1.6 to 1)

- **Translation**
  A pedestrian can be detected not only in the center of the bounding box, but also on the sides. By providing strictly images of pedestrians estimated in the middle of the bounding box image the network will tend to "draw" continuously a head in that location in test time. Therefore it is also of importance to provide a variation in translation to the network. This is done by translating each pedestrian bounding box in four direction with respect to the bounding box size: 30% left, 30% right, 30% upwards and 30% left and upwards combined. The lower 30% translation seems insignificant given the procedure that will be explained in the online module.

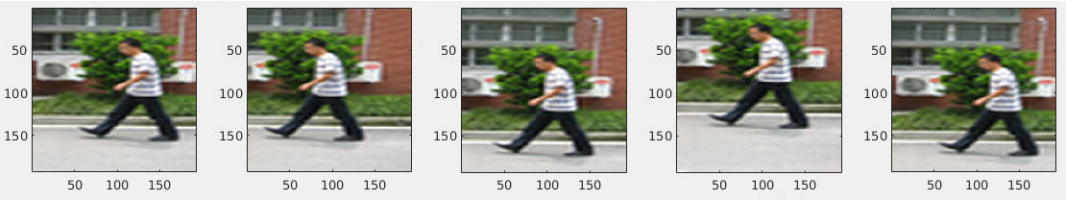

Figure 5.8: An example of a training sample translated at the five different directions on a 1.6 scale.

- **Rotation**
  Rotation, although not as significant, can also increase the number of data. Pedestrians are usually not represented rotated but a rotation of -7 and +7 degrees keeps rotation invariance and at that same time is a realistic scenario (in an example where the camera could be slightly shifted).

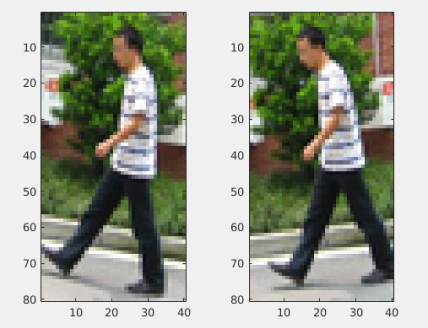

Figure 5.9: An example of a training sample rotated at the two different angles.

- **Flipping**
  Flipping is a very useful augmentation method to easily double the data size. Therefore all above augmentation techniques are also flipped.

5.1.4 Network Training Parameters

The basic parameters of the network are worth discussing, since they are of importance in order for efficient training. They include the following:

- **Learning Rate**
  The learning rate of the network is set to 10^{-9}. When dealing with fully convolutional neural networks learning rates are set lower than in usual networks since prediction happens in a more detailed level, and therefore a lower learning rate will assure a "smoother" transition of the weights towards the goal.

- **Overall Iterations**
  The network is learned for 100k iterations. This number is chosen as a trade off for both the small learning rate (since the network is learning slower it will also need more iterations) and in order to avoid overfitting. In general, convolutional neural networks when finetuning tend to train for approximately 80k iterations.

- **Loss iterations**
  The average loss is reported every 20 iterations. Reporting the loss in less iterations would harm the network since from image to image this may vary largely, so an average over 20 iterations will provide more stability.

- **Test interval**
  For every 500 iterations, the validation set is used in order to evaluate the network. The loss also in this set is reported and used to adjust the weights. The validation period lasts for 500 iterations and then training is resumed.

- **Mean Subtraction**
  The proposed network is pre-trained on the icome dataset for cloth parsing. This pre-trained network deals well with getting an overall definition of the location of a person, but deals poorly with getting details and with low resolution. Since this is the dataset used for pre-training, the mean image of the dataset is computed and for each training and test image this mean image is subtracted. Mean subtraction tends to center the data along every dimension around the origin, thereby increasing accuracy.

- **Weight Initialization**

  The weights of the network need to be initialized before training begins, otherwise there is danger in the network not being able to learn anything since all backpropagation parameters will be equal. The convolutional layers are initialized with a gaussian distribution where the de-convolutional layers use the xavier initialization ([21]).

5.2 Online

The following section describes the online module, which describes the way in which the network is used in test time in order to evaluate new data.

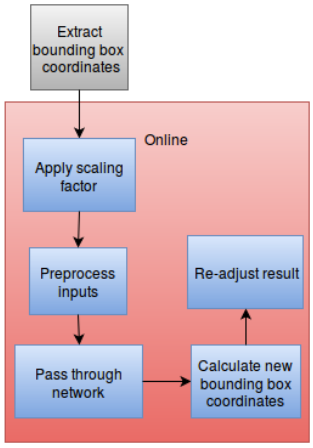

Figure 5.10: System design (online module)

5.2.1 Extract bounding box coordinates

In test time, the network relies on a pedestrian detector in order to receive an initial estimation. Usually detectors receive a whole image where pedestrians will be located at some regions (such as in Figure 1.1). These cropped regions are then used in order to evaluate the detector result.

5.2.2 Apply scaling factor

The assumption is made that the detector may not perform well in certain situations. Therefore, a scaling factor is applied to every bounding box in order to assure that if the measurement if incorrect, there will be a margin that will assure that the pedestrian is included in the area. The scaling factor should not be too large but not too small as well, since in the prior case it can not provide the whole pedestrian in size where in the latter case there can be too much background in the bounding box which will add too much noise to the measurement.

5.2.3 Pre-process inputs/Pass through network

Once the scaling factor is applied, the procedure to pass the image through the network begins. The image is preprocessed by applying the resize method describe in Subsection 5.1.2, subtracting the mean value from the icome dataset and converting the image to single value precision. The network returns two 192x192 heatmaps for the foreground and background estimation. These are then fused in order to return the final result, which is a binary image where 1 represents a pedestrian and 0 the background. The pixel decision is made by looking at which channel has the highest value for the given pixel.

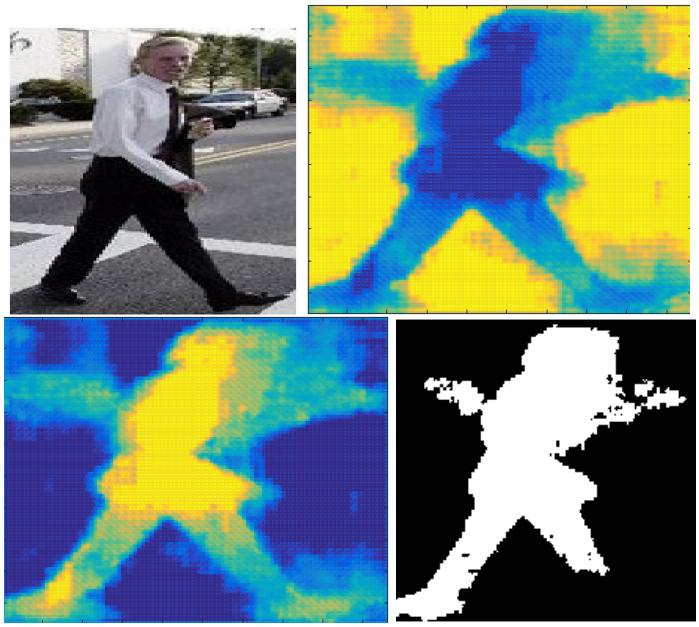

Figure 5.11: Four stages of pedestrian image. Top left:input image. Top Right:Background Heatmap. Lower left: Foreground Heatmap. Lower right: Fused Result. [39]

5.2.4 Calculate new bounding box coordinates/Re-adjust result

After the binary image is constructed, the final refined estimation is made. It should be expected that there will be an amount of noise in this estimation so therefore only the biggest BLOB is kept. From this BLOB, the coordinates of the bounding box are measured with respect to three spaces: the 192x192 network estimation, the original bounding box (before the resize) and the global input image. This way the new result can be plotted on the image and compared with ground truth/detector estimation.

Chapter 6

Results

The following chapter evaluates the proposed network on a series of tests. First, the results of the network in the offline module are discussed, where various plots and images of the network estimation are described. In the second part, the online evaluation of the network is analyzed on a series of pedestrian benchmarks(Section 4.5). In these, two different tests will be made: an evaluation of the network against state-of-the-art detectors will be made in the first test. In the second test, overlap precision will be measured between the ground truth and a detector with and without the network on a series of datasets. The chapter ends with an analysis of the success and failure cases. Each measurement is implemented in MATLAB and run on a server with a Tesla K40c GPU (12GB).

6.1 Offline

Training Results

As mentioned in Chapter 3, as the number of iterations increase the loss should decrease. It is also of interest to look at the shape of the curve when plotting iterations versus train loss. In an ideal situation, the curve's shape should have an exponential decrease ([26]). This means that the chosen learning rate will contribute efficiently in decreasing the loss. If the curve has an exponential decrease in the first thousand iterations but then stabilizes or starts rising, then the learning rate is too high and the network cannot learn through time. The plot of the proposed network for both training and validation can be seen at Figure 6.1 and Figure 6.2.

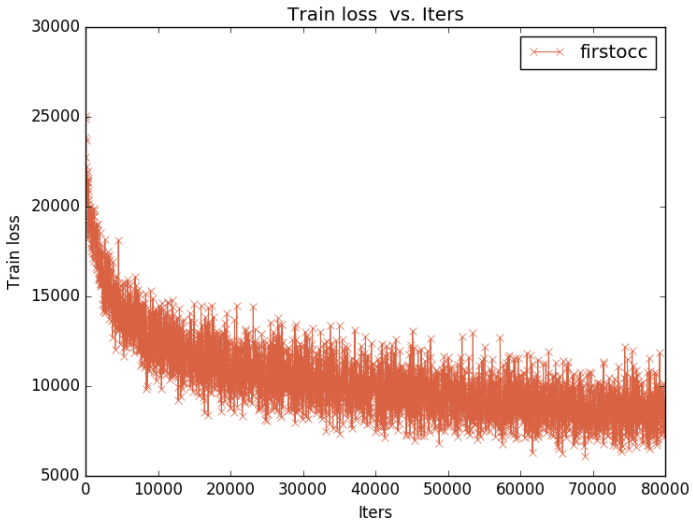

Figure 6.1: Plot of network training loss versus iteration.

Figure 6.2: Plot of network validation loss versus iteration.

By observing the plots it can be noticed that even though the loss decreases through time, it fluctuates through iterations. This relatively small fluctuation should be expected, since the weights continually attempt to adjust to the training samples, which always differ to each other to some extent. Another reason to this behaviour is that pixel-wise classification is detailed so a small difference in error between two approximations can still increase/decrease the loss.

The training loss is measured in thousands. This number denotes the sum of losses accross the spatial coordinates of the label. The mean Intersection over Union (true positive / (true positive + false positive + false negative)) was founded to be at 0.55 on the validation set, which achieves state-of-the-art([32]).

What does the network learn?

Through training, the network should progressively learn a pedestrians representation. Since the network is first trained on the icome dataset, a simple comparison of the outputs is made between the network before and after finetuning with the proposed dataset (Figure 6.3).

Figure 6.3: From left to right:Input Image, output from pre-trained icome dataset, output from finetuned on the proposed dataset.

The pre-trained network does not segment the pedestrian in a logical way. This can be interpreted by the fact that the icome dataset contains high resolution images and not only of pedestrians (e.g. people with varying poses, people sitting down).

Occlusion Handling

Occlusion is expected to be handled by inserting the manual labeled occluded dataset, therefore it is of interest to see its effect on partially occluded pedestrians. A figure illustrating this procedure can be shown at Figure 6.4.

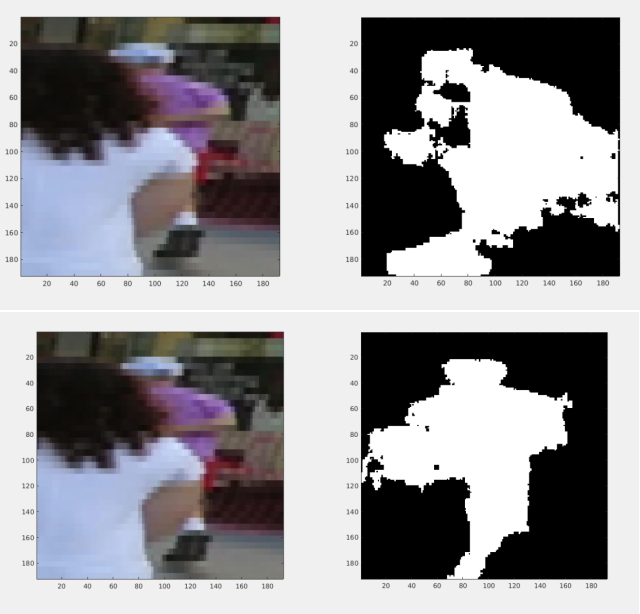

Figure 6.4: Top image: network output after 80k iterations without the occlusion dataset. Bottom image:network output after 80k iterations with the occlusion dataset.

Although the bottom figure still contains some noise, the approximation is significantly more precise. By adding partial occluded images, the network learns that it is not always the case that after the pedestrian head a body will occur and legs.

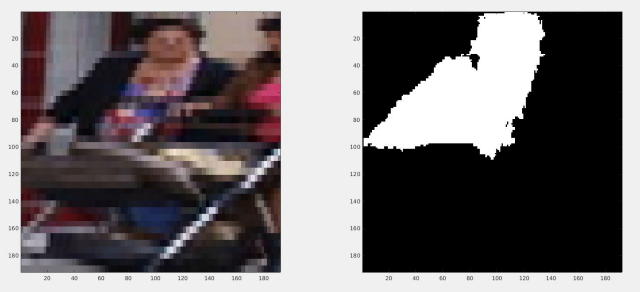

Figure 6.5: Occluded pedestrian by external object

6.2 Online

The online testing regards examining the networks performance on a series of measurements and comparing it to existing methods. For each test, the Aggregated Channel Features detector ([9]) is used (trained on the INRIA dataset) as the basis for the network.

6.2.1 Comparison with State-of-the-art

The proposed networks goal is to refine a detectors result, therefore it is of importance to compare its performance with state-of-the-art methods. The following test examines the networks performance on the datasets mentioned in Section 4.5. First the setup is discussed and afterwards the log-log plots are shown.

Test Setup

The setup in [11] is adopted, where log-average miss rate versus false positives per image is plotted by thresholding the confidence score. The procedure is the same for each dataset, where each image is passed through the detector which returns a number of pedestrian locations. Each pedestrian location is passed through the network and its refined result is computed as mentioned in Section 5.2. These detections are then attempted to be filtered out, resulting to one detection per ground truth label. Detections with the highest confidence score are matched first since they are most likely to be good measurements.

A detection is matched if its overlap precision is bigger than 0.5. This metric measures the overlap between the ground truth and the algorithm estimation. Specifically, the overlap denotes the intersection union between the two regions (Equation 6.1).

$$a_o = \frac{area(BB_{dt} \bigcap BB_{gt})}{area(BB_{gt} \bigcup BB_{gt})} \tag{6.1}$$

The overlap is important in order to correctly evaluate precision of detectors. Furthermore, it can also be used as a factor to filter out noisy measurements. For example, in [15], a correct detection is considered one where the overlap exceeds 0.5 (where 1 is a perfect overlap).
If a given detection result matches more than one ground truth label, the one with the highest overlap precision is kept as a match for that label. For more details, the readers are advised to go through the original publication that plots these results ([11]).

Segmentation Network confidence score

As mentioned above, the confidence score for each result is crucial in order to compare to state-of-the-art methods, since it is used for thresholding the miss rate/false positive plot and also to filter out detections. Therefore for each networks result there should be an approximation of confidence. Since the proposed network does not return a confidence score for its refined results, the Aggregated Channel Features detector trained on the Caltech dataset is used on each result and the resulting

confidence is used for evaluation. The reason behind using the Caltech trained detector for this task is due to the fact that the sliding window size that the authors chose when training is smaller than the INRIA one (64x32 versus 128x64). It is also worth noting that the detector is trained on full-scene images and not cropped bounding boxes. For this reason each bounding box is enlarged by a scale factor of 1.2 in order to increase confidence precision, but still being close to the networks result and add the minimum ammount of noise.
Since the detector is trained to approximate location and confidence, it will still return a confidence for a given estimated bounding box. This confidence does not describe the networks approximation though (for example,if the networks bounding box contains background the confidence will describe the enclosed pedestrian detectors approximation rather than the whole bounding box). For this reason the percentage that the detectors bounding box covers in the networks output is computed and that percentage of the confidence is kept. For example, if the networks output is a 200x200 image and the detector applied on the scaled image returns a 100x100 bounding box with confidence of 100, the resulting confidence is will be 50 since the detectors approximation covers half the networks approximation. It should be noted that cases exist where the detector approximation is larger than the network output. In theory this could lead to an unrealistic confidence score, although in practice this does not happen since the scaling factor is so low. An example can be shown in Figure 6.6.

Figure 6.6: Confidence approximation for network result. Left: Network output. Right: Scaled bounding box. The black bounding box denotes the detectors approximation. The initial confidence is 68.19 and the final confidence is 66.9.

It is also worth noting that in some cases the returned approximation from the network is smaller than the detector window, which results into the detector not being able to be applied. In this case the approximation is most likely a false

positive or a low resolution image. In both cases, the output will not be a good approximation of the pedestrian and therefore the confidence is set to zero.

Results

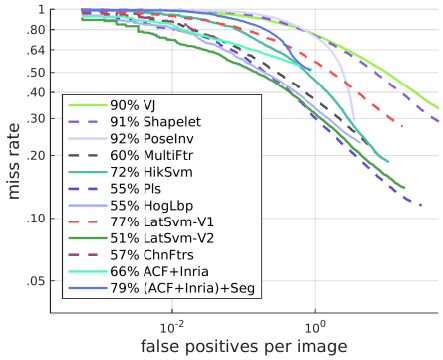

Figure 6.7: ETH dataset. Proposed method (ACF+Inria)+Seg against detector without network ACF+Inria ([9]) and state-of-the-art detectors Viola Jones([42]), Shapelet([38]), PoseInv([30]), MultiFtr([47]), HikSvm([34]), Pls([40]), HogLbp([45]) , latsvm-v1 [18], latsvm-v2 [19] and ChnFtrs [10].

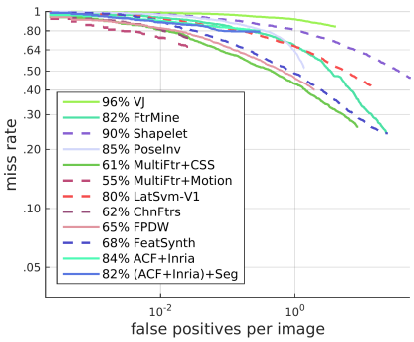

Figure 6.8: Caltech dataset. Proposed method (ACF+Inria)+Seg against detector without network ACF+Inria ([9]) and state-of-the-art detectors Viola Jones([42]), FtrMine ([6]), Shapelet([38]), PoseInv([30]), MultiFtr+CSS ([43]), MultiFtr+Motion ([43]), latsvm-v1 ([18]), ChnFtrs ([10]), FPDW ([8]) and FeatSynth ([1]).

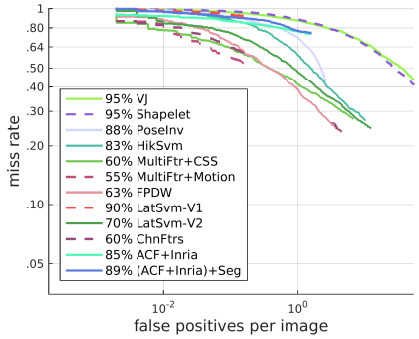

Figure 6.9: Tud-Brussels dataset. Proposed method (ACF+Inria)+Seg against detector without network ACF+Inria ([9]) and state-of-the-art detectors Viola Jones([42]), Shapelet([38]), PoseInv([30]), HikSvm([34]), MultiFtr+CSS ([43]), MultiFtr+Motion ([43]), FPDW ([8]), LatSvm-V1 ([18]), LatSvm-V2 ([19]) and ChnFtrs ([10]).

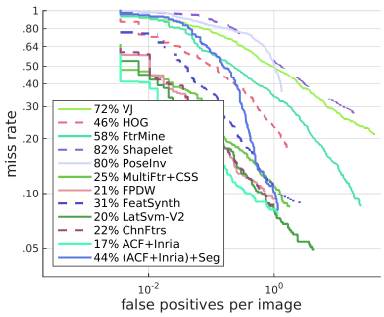

Figure 6.10: INRIA dataset. Proposed method (ACF+Inria)+Seg against detector without network ACF+Inria ([9]) and state-of-the-art detectors Viola Jones([42]), FtrMine ([6]), Shapelet([38]), HOG([4]), PoseInv([30]), MultiFtr+CSS ([43]), FPDW ([8]), LatSvm-V2 ([19]) and ChnFtrs ([10]).

For each dataset, the results of the five best and five worst detectors in terms of performance are used for comparison. Furthermore, performance of the ACF detector without the proposed network is measured (ACF+Inria) and of the ACF detector with the segmentation network ((ACF+Inria)+Seg). The proposed network can achieve state-of-the-art performance over all datasets. There is a small decline in average miss rate in the caltech dataset (2%) where in the other three datasets there is a small increase of 4%-7% except for the INRIA dataset (more justification on this can be found in the next section).

Furthermore, results from the caltech dataset on occlusion and scale factors can be seen at Figure 6.18 and Figure 6.14. Scale is divided into three categories : near (80 or more pixels), medium (30-80 pixels) and far (30 pixels or less). Occlusion is divided into No Occlusion(0% occluded area), Partial occlusion (1-35% occluded area) and heavy occlusion (36-80% occluded area).

The network can deal with occlusion and performance is not affected. Furthermore in no and partial occlusion a small improvement of 2% is noted. All detectors suffer from small size pedestrians (far scale) and therefore the network cannot improve performance. Medium scale is also a difficult to improve area, since the average height is significantly smaller than the networks input (192 versus 30-80). On near scales the segmentation network shows an improvement of 4% in comparison to the detector itself.

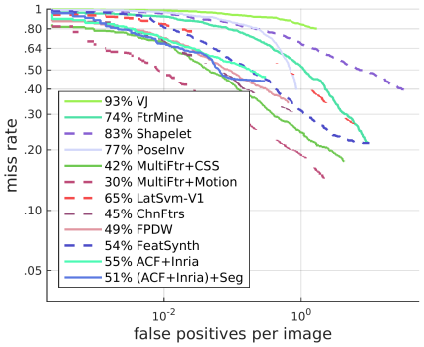

Figure 6.11: Caltech dataset (Near Scale). Proposed method (ACF+Inria)+Seg against detector without network ACF+Inria ([9]) and state-of-the-art detectors Viola Jones([42]), FtrMine ([6]), Shapelet([38]), PoseInv([30]), MultiFtr+CSS ([43]), MultiFtr+Motion ([43]), latsvm-v1 ([18]), ChnFtrs ([10]), FPDW ([8]) and FeatSynth ([1]).

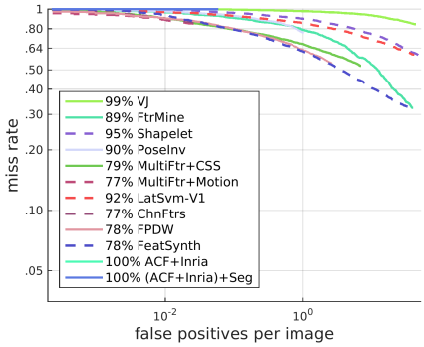

Figure 6.12: Caltech dataset (Medium Scale). Proposed method (ACF+Inria)+Seg against detector without network ACF+Inria ([9]) and state-of-the-art detectors Viola Jones([42]), FtrMine ([6]), Shapelet([38]), PoseInv([30]), MultiFtr+CSS ([43]), MultiFtr+Motion ([43]), latsvm-v1 ([18]), ChnFtrs ([10]), FPDW ([8]) and FeatSynth ([1]).

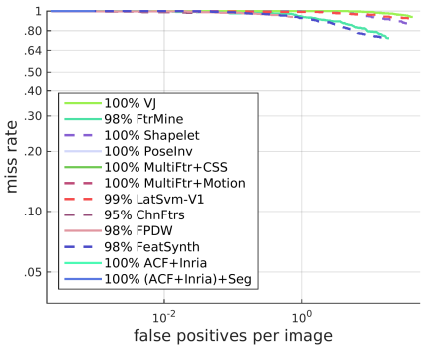

Figure 6.13: Caltech dataset (Far Scale). Proposed method (ACF+Inria)+Seg against detector without network ACF+Inria ([9]) and state-of-the-art detectors Viola Jones([42]), FtrMine ([6]), Shapelet([38]), PoseInv([30]), MultiFtr+CSS ([43]), MultiFtr+Motion ([43]), latsvm-v1 ([18]), ChnFtrs ([10]), FPDW ([8]) and FeatSynth ([1]).

Figure 6.14: Caltech results on pedestrians of various scales

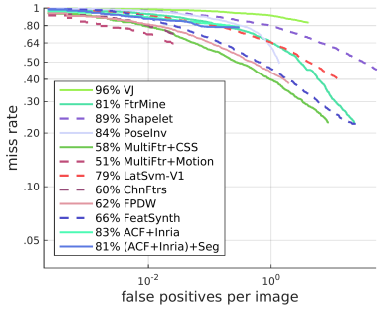

Figure 6.15: Caltech dataset (No Occlusion). Proposed method (ACF+Inria)+Seg against detector without network ACF+Inria ([9]) and state-of-the-art detectors Viola Jones([42]), FtrMine ([6]), Shapelet([38]), PoseInv([30]), MultiFtr+CSS ([43]), MultiFtr+Motion ([43]), latsvm-v1 ([18]), ChnFtrs ([10]), FPDW ([8]) and FeatSynth ([1]).

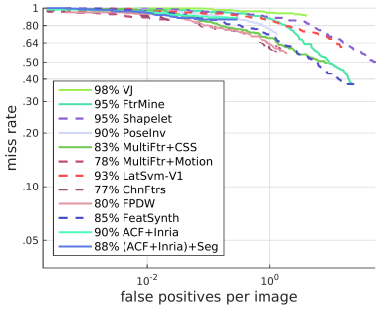

Figure 6.16: Caltech dataset (Partial Occlusion). Proposed method (ACF+Inria)+Seg against detector without network ACF+Inria ([9]) and state-of-the-art detectors Viola Jones([42]), FtrMine ([6]), Shapelet([38]), PoseInv([30]), MultiFtr+CSS ([43]), MultiFtr+Motion ([43]), latsvm-v1 ([18]), ChnFtrs ([10]), FPDW ([8]) and FeatSynth ([1]).

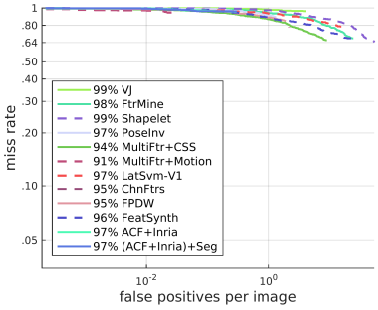

Figure 6.17: Caltech dataset (Heavy Occlusion). Proposed method (ACF+Inria)+Seg against detector without network ACF+Inria ([9]) and state-of-the-art detectors Viola Jones([42]), FtrMine ([6]), Shapelet([38]), PoseInv([30]), MultiFtr+CSS ([43]), MultiFtr+Motion ([43]), latsvm-v1 ([18]), ChnFtrs ([10]), FPDW ([8]) and FeatSynth ([1]).

Figure 6.18: Caltech results on pedestrian of various occlusion

6.2.2 Segmentation Error/Scaling Factor Test

The following test evaluates the error between the aggregated channel features detector performance and the detector when adding the proposed network. Addi-

tionaly, the scale factor that gives the highest performance for the network in each dataset is studied along with a number of additional properties such as false positives and recoveries (where the network returns higher accuracy than the network itself).

Test Setup

The test is conducted on the same datasets as in Sub-Section 6.2.1. For each image in a given dataset, a number of bounding boxes are returned from the detector. Similarly to Sub-Section 6.2.1, false positives and duplicate detections are filtered out, by matching each ground truth label to a detector result. The networks refined result is then evaluated for each bounding box as explained in Section 5.2. The two bounding boxes overlap precision with respect to the ground truth is then evaluated.

The measure of the segmentations performance error versus the scaling factor is plotted. The networks performance should be expected to vary when applying different scale factors so therefore it is of interest to observe its affect to the overall performance. Segmentation error is measured by subtracting the average overlap precision of all detected pedestrians of the two methods (Equation 6.2).

$$E = \frac{\sum_1^i ovrD_i}{P} - \frac{\sum_1^i ovrS_i}{P} \tag{6.2}$$

Where E denotes the segmentation error, overD the overlap ratio of the detector for the i-th pedestrian, overS the overlap ratio of the network for the i-th pedestrian and P the total number of pedestrians.

Results

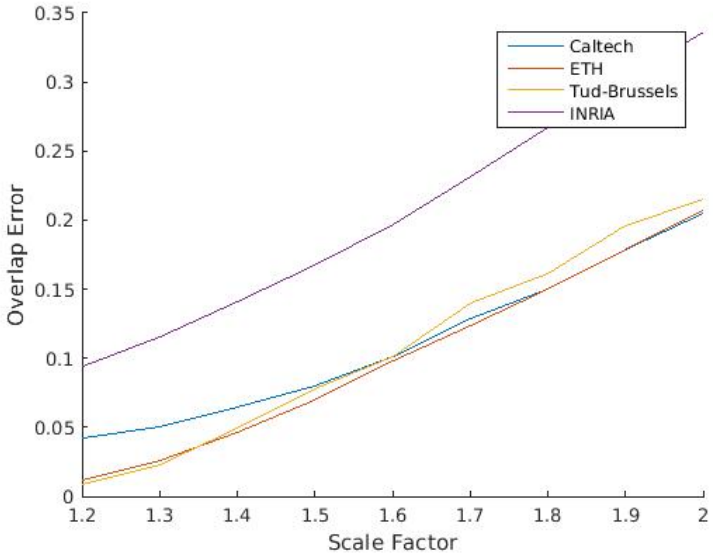

Figure 6.19: ScaleFactor versus Precision Overlap error for each dataset.

The general trend of the results show that smaller scale factors achieve better performance. This can be justified from the fact that only detection results with an overlap higher than 0.5 were kept for evaluation, so the measurements in general were already good. Keeping this into consideration, a re-evaluation of a >0.5 overlaping bounding box with a scalefactor of 1.2 or 1.3 giving optimal results can be considered a valid argument.

Additional information about the results on the lowest scale factor (1.2) can be seen at Table 6.1.

Dataset	Seg. Overlap	Det. Overlap	Hits	Misses	False Positives
ETH	0.7562	0.7683	2543	2781	115
Caltech	0.6946	0.7371	128	230	24
Tud-Brussels	0.7138	0.7225	112	123	5
INRIA	0.7525	0.8467	324	1374	32

Table 6.1: Additional results on segmentation error result

Where Seg.Overlap denotes segmentation overlap precision, Det.Overlap detector overlap precision, hits the number of bounding boxes the segmentation algorithm scores a higher overlap than the detector, misses the number of bounding

boxes the detector scores a higher overlap than the segmentation algorithm and false positives the number of bounding boxes that the detector scores higher than 0.5 and the segmentation scores lower than 0.5. It is noted that the average time in seconds taken for the network to process an image was found to be 0.56 seconds.

By looking at the results,it can be noted that the INRIA dataset shows a higher error than the other datasets. The reasoning behind this is that the ground truth labeled in [11] which was used for evaluation is constructed by standarizing the aspect ratio in the pedestrians width, in order to remove the effect limbs and pose variations have on benchmarking. The proposed network does not standarize aspect ratio though so its performance is more effectively measured when using enclosed bounding boxes. Although for the ETH and Tud-Brussels dataset the ground truth labels from the original authors were used ([14], [46]), the INRIA dataset in general contains images of pedestrians and non-pedestrians and re-arranging images and ground truth labels is out of the scope of this project. Furthermore, the results of the other datasets can show the effectiveness of the proposed network. An example of this issue can be shown at Figure 6.20.

Figure 6.20: Standarized aspect ratio ground truth labeling for INRIA dataset. Red box denotes ground truth, blue denotes detector result and green denotes segmentation result

6.2.3 Analysis of Results

In the previous sections a series of testing were made in order to evaluate the networks performance. It has been seen that results vary to a certain extent and therefore in this section the results are analyzed and the factors that degrade per-

formance are examined.

Confidence Score

As mentioned in Sub-Section 6.2.1, the confidence score for the network in the comparison with the state of the art is calculated by applying a pre-trained detector on the resulting bounding boxes of the network. Due to the detector being trained on full scene images and not on bounding boxes this approximation in some cases is unstable, even though a detector with a relatively small window size is used. Ideally a confidence approximation that is constructed specifically for bounding boxes would yield optimal results. It should be noted that the detector applied is not significantly unstable in most cases and therefore does not affect the confidence score in a major degree. An example that illustrates this behaviour can be seen at Figure 6.21.

Figure 6.21: Confidence score error. The left image results in a confidence of 17.85 where the right image results to a confidence of 165.10. Both images enlose tightly a pedestrian and therefore should yield a relatively high score (as in the right image).

Ammount of training data

The amount of training data is relatively small as mentioned in Section 4.5. Even though an attempt has been made to compensate for this through data augmen-

tation and manual labeling, there is still room for improvement. For example, in [41], approximately 1.3 million images are used for training, 50k for validation and 100k for training. Even though the proposed network (combining the pre-trained and the fine-tuned dataset) is consisted of far more less data, it can still achieve reasonable results, even though there exist cases where results vary.

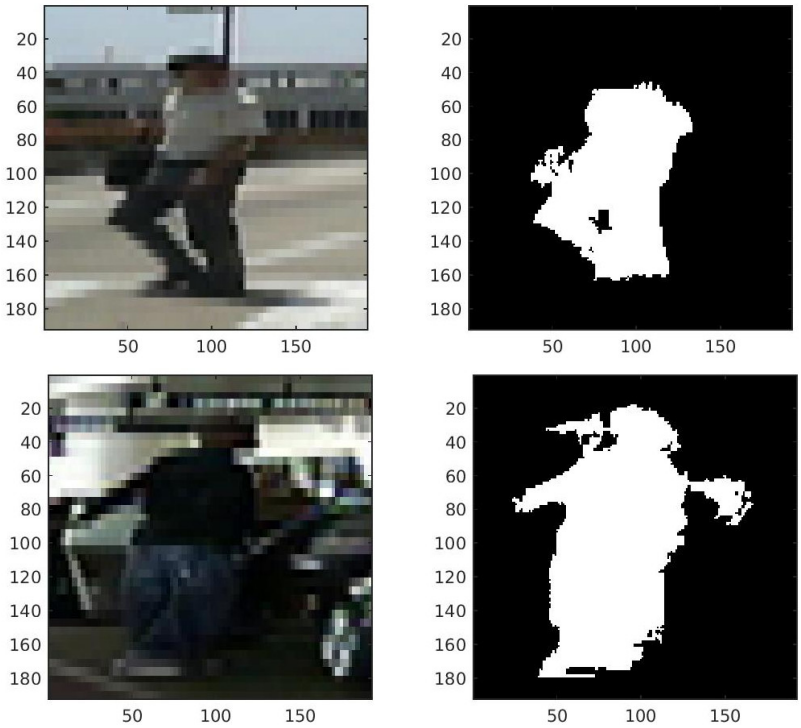

Figure 6.22: Measurement which shows how the lack of training data can affect performance (from the caltech dataset). The upper sample denotes a sample which is not properly segmented where the lower denotes one that is properly segmented.

Background Clutter

Background clutter (as mentioned in Sub-Section **??**) is a fundamental problem in detection and segmentation and therefore it should be expected to be a factor that decreases performance to a certain extent. Measurements show that the network is relatively sensitive to background clutter, meaning that if the background at some areas close to the pedestrian are alike in values then they will be segmented as well. This has been mostly detected close to the area of the head.

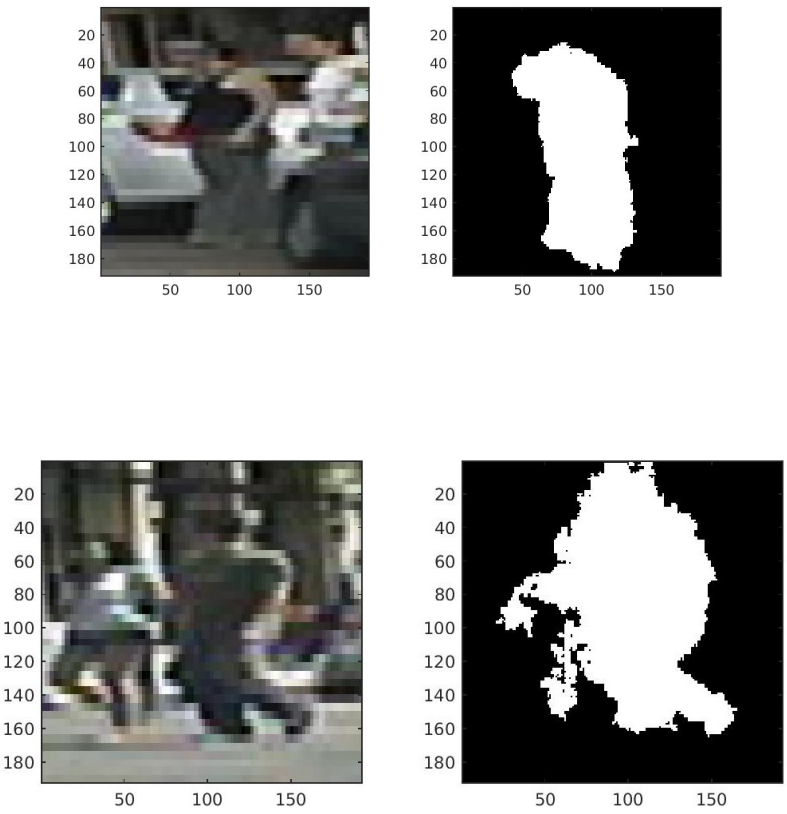

Figure 6.23: Measurement which shows how background clutter can affect performance (from the caltech dataset). The upper sample denotes a sample which the lower area of the pedestrian and the background have similar texture, where the lower denotes the same case for the upper area.

Pose/Accesory approximation

Chapter 2 discusses how pose estimation and different acccessories can affect performance since these factors can not be easily predictable and can vary to a wide extent. The occlusion dataset seems to make a good approximation of when an external object is present. Furthermore, it can also segment pedestrians in various poses other than walking.

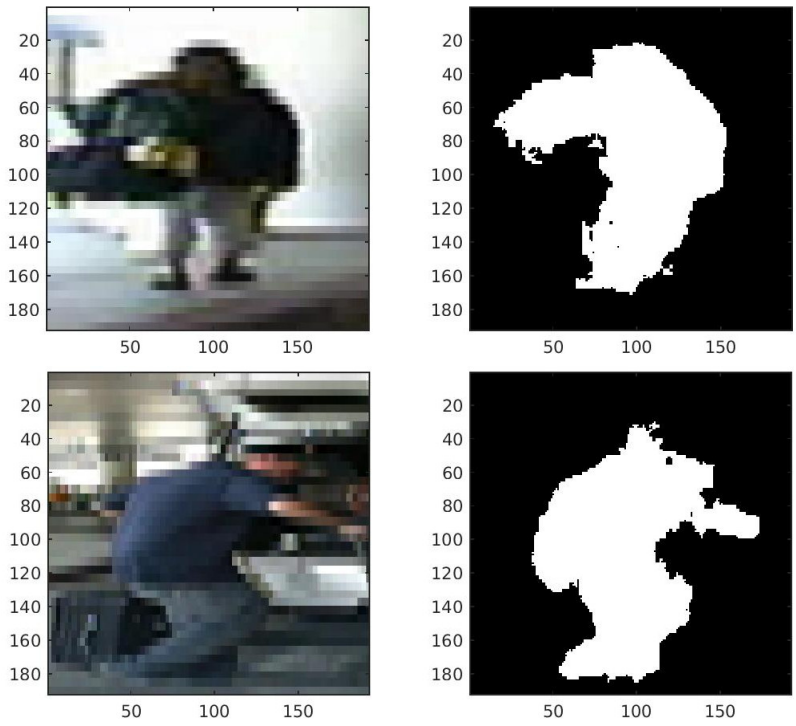

Figure 6.24: Measurement which shows accesory (upper image) and pose (lower image) approximation (from the caltech dataset).

Multiple pedestrians

The training data used for the network at most cases include a single pedestrian and in the cases where they include more than one only one is labeled. This results in confusion during test time if there are more than one pedestrian in the image since the network does not know where to focus the segmentation on and the end result will be a segmentation of multiple pedestrians. This issue could be solved by creating a network that can identify single or multiple pedestrians and assign a separate bounding box for each case (therefore the training labels would be from 0-K, where K is the number of pedestrians present in each image). Two cases that show this issue can be seen at Figure 6.25.

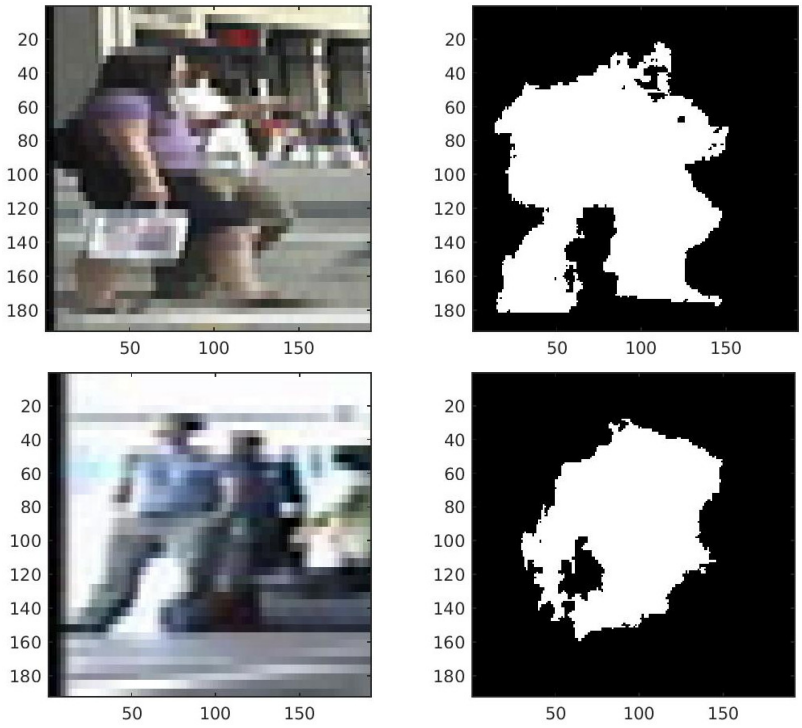

Figure 6.25: Measurement which shows how the presence of more than one pedestrian can affect performance from the caltech dataset. The result is a combination of the two pedestrians present in the image.

Chapter 7

Conclusion

Throughout this master thesis, the problem of detecting failures in pedestrian detection and refining their results has been approached. For this problem, a fully convolutional neural network has been constructed which attemps to classify an image on a pixel level and detect the size, shape and location of the pedestrian. In order to train this network, various datasets have been used and altered in order to fit the cause. Furthermore, in order to deal with pedestrians that are partially occluded, a manual partial occluded dataset has been constructed. By observing the training of the network through time it can be proven that the network can learn a pedestrian representation and segment one in many cases and can achieve state-of-the-art performance in terms of mean Intersection over Union over classes (0.55).

The proposed network was combined with an object detector and tested with two test approaches. The first approach includes comparing the proposed network with the state-of-the-art methods under a number of famous benchmarks in the pedestrian detection area. By thresholding a computed confidence score , the log-average miss rate was calculated over false-positives per image. Results show that in most cases the proposed network shows a minor decrease (2-7%) on average, although in some situations it does improve performance (e.g. occlusion). In situations where the combined detector performance is weak the proposed network is not able to perform better (e.g. heavy occlusion).

The second testing approach deals with comparing results of bounding boxes overlap between ground truth labels and detector results with and without the proposed network. For each detector result, the network is applied and the two results are compared with the ground truth label. Results show that even though the proposed network can improve results on certain occasions, there still exist many cases where the detector outperforms the network and the addition of the network harms performance by a overlap difference of 0.01-0.03.

Furthermore an analysis of why the network harms performance is made. This includes a number of reasons including lack of sufficient amount of training data and challenging factors (background clutter, multiple pedestrians in image, confidence estimation).

Appendix A

Labeling program

For the custom partial occluded dataset made, a custom python program for labeling objects is used. The program was initially made for labeling cars, but is also applicable for pedestrians. Polygons are created by saving each point the user clicks on the image selected. An example of the GUI can be seen at Figure A.1.

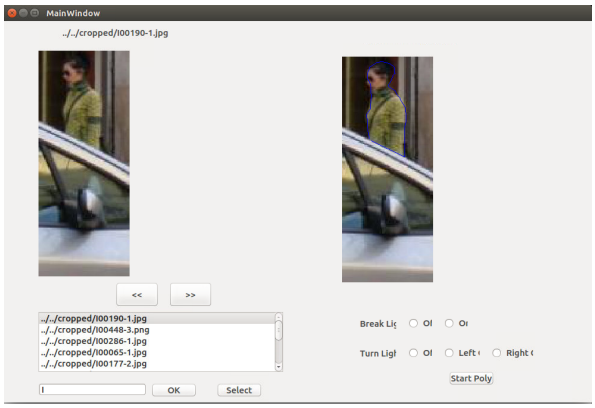

Figure A.1: Labeling softwared gui. The blue polygon denotes points clicked by the user

Once the key **Z** is pressed, the points are saved and the button **start poly** saves a .txt file with the coordinates for the given image. These coordinates are then used in order to create the mask of the pedestrian in MATLAB:

Figure A.2: Output of constructing label with polygon coordinates

Bibliography

[1] Bar-Hillel Aharon et al. "Computer Vision – ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV". In: 2010. Chap. Part-Based Feature Synthesis for Human Detection.

[2] P. Arbeláez et al. "Multiscale Combinatorial Grouping". In: *Computer Vision and Pattern Recognition*. 2014.

[3] T.F. Cootes et al. "Active Shape Models-Their Training and Application". In: *Computer Vision and Image Understanding*. 1995.

[4] Navneet Dalal and Bill Triggs. "Histograms of Oriented Gradients for Human Detection". In: *International Conference on Computer Vision & Pattern Recognition*. Ed. by Cordelia Schmid, Stefano Soatto, and Carlo Tomasi. Vol. 2. 2005, pp. 886–893. URL: http://lear.inrialpes.fr/pubs/2005/DT05.

[5] Sander Dieleman, Kyle W. Willett, and Joni Dambre. *Rotation-invariant convolutional neural networks for galaxy morphology prediction*. http://mnras.oxfordjournals.org/content/450/2/1441. 2015.

[6] P. Dollar et al. "Feature Mining for Image Classification". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007.

[7] Piotr Dollar et al. *Pedestrian Detection: A Benchmark*. http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/files/CVPR09pedestrians.pdf. 2009.

[8] Piotr Dollár, Pietro Perona, and Serge Belongie. *The Fastest Pedestrian Detector in the West*. https://www.robots.ox.ac.uk/~vgg/rg/papers/DollarBMVC10FPDW.pdf. 2010.

[9] Piotr Dollár et al. *Fast Feature Pyramids for Object Detection*. http://research-srv.microsoft.com/pubs/220572/DollarPAMI14pyramids.pdf. 2014.

[10] Piotr Dollár et al. *Integral Channel Features*. http://pages.ucsd.edu/~ztu/publication/dollarBMVC09ChnFtrs_0.pdf. 2009.

[11] Piotr Dollár et al. *Pedestrian Detection: An Evaluation of the State of the Art*. https://pdfs.semanticscholar.org/fc8f/99bd98f7c8f194e9ac6cd62b565aa8155935.pdf. 2012.

[12] A. Ess, B. Leibe, and L. Van Gool. *Depth and Appearance for Mobile Scene Analysis*. https://data.vision.ee.ethz.ch/cvl/aess/iccv2007/. 2007.

[13] A. Ess et al. *A Mobile Vision System for Robust Multi-Person Tracking*. https://data.vision.ee.ethz.ch/cvl/aess/cvpr2008/. 2008.

[14] Andreas Ess et al. *A Mobile Vision System for Robust Multi-Person Tracking*. ftp://ftp.vision.ee.ethz.ch/publications/proceedings/eth_biwi_00543.pdf. 2008.

[15] Mark Everingham et al. *A High Accuracy Pedestrian Detection System Combining a Cascade AdaBoost Detector and Random Vector Functional-Link Net*. http://homepages.inf.ed.ac.uk/ckiw/postscript/ijcv_voc09.pdf. 2014.

[16] Dariu Gavrila (Daimler) Fabian Flohr (Daimler). "PedCut: an iterative framework for pedestrian segmentation combining shape models and multiple data cues". In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2013.

[17] P. Felzenszwalb and D. Huttenlocher. "Efficient Graph-Based Image Segmentation". In: *International Journal of Computer Vision*. 2004.

[18] P. Felzenszwalb, D. McAllester, and D. Ramanan. "A discriminatively trained, multiscale, deformable part model". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. 2008.

[19] P. F. Felzenszwalb et al. "Object Detection with Discriminatively Trained Part-Based Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2010).

[20] D. M. GAVRILA and S. MUNDER. *Multi-cue Pedestrian Detection and Tracking from a Moving Vehicle*. http://www.gavrila.net/ijcv07_springer.pdf. 2007.

[21] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*. 2010.

[22] Bharath Hariharan et al. "Simultaneous Detection and Segmentation". In: *European Conference on Computer Vision (ECCV)*. 2014.

[23] Heiko Hirschmuller. "Stereo Processing by Semiglobal Matching and Mutual Information". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2008.

[24] Marín J et al. *Occlusion Handling via Random Subspace Classifiers for Human Detection*. https://pdfs.semanticscholar.org/fc8f/99bd98f7c8f194e9ac6cd62b565aa8155935.pdf. 2014.

[25] Z. Kalal, K. Mikolajczyk, and J. Matas. "Forward-Backward Error: Automatic Detection of Tracking Failures". In: *Pattern Recognition (ICPR), 2010 20th International Conference on*. 2010, pp. 2756–2759. DOI: 10.1109/ICPR.2010.675.

[26] Andrej Karpathy, Justin Johnson, and Fei-Fei Li. *Convolutional Neural Networks for Visual Recognition*. http://cs231n.stanford.edu/index.html. 2015.

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

[28] Kasper Skou Ladefoged et al. *Pedestrian Counting Using Stereo Thermal Cameras*. http://projekter.aau.dk/projekter/files/213063899/15gr840.pdf. 2014.

[29] A Li et al. "NUS-PRO: A New Visual Tracking Challenge". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2 (2016), pp. 335–349.

[30] Zhe Lin and Larry S. Davis. *A Pose-Invariant Descriptor for Human Detection and Segmentation*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.154.7771&rep=rep1&type=pdf. 2008.

[31] localfr. *Paris: New plan to hand centre to pedestrians*. http://www.thelocal.fr. 2014.

[32] Jonathan Long, Evan Shelhamer, and Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. http://www.cs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf. 2012.

[33] Yalong Ma et al. *Pedestrian Detection and Tracking from Low-Resolution Unmanned Aerial Vehicle Thermal Imagery*. www.mdpi.com/1424-8220/16/4/446/pdf. 2016.

[34] Subhransu Maji, Alexander C. Berg, and Jitendra Malik. *Classification using intersection kernel support vector machines is efficient*. http://acberg.com/papers/mbm08cvpr.pdf. 2008.

[35] Ofer Matan et al. "Multi-Digit Recognition Using A Space Displacement Neural Network". In: *Neural Information Processing Systems*. 1992.

[36] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. *Learning Deconvolution Network for Semantic Segmentation*. http://arxiv.org/pdf/1505.04366v1.pdf. 2015.

[37] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[38] Payam Sabzmeydani and Greg Mori. *Detecting Pedestrians by Learning Shapelet Features*. https://www.cs.sfu.ca/~mori/research/papers/sabzmeydani_shapelet_cvpr07.pdf. 2007.

[39] Katherine Santiago. *Pedestrian decoy program begins in southern N.J. communities*. http://www.nj.com/news/index.ssf/2009/08/nj_traffic_safety_officials_st.html. 2009.

[40] William Robson Schwartz et al. *Human Detection Using Partial Least Squares Analysis*. http://www.umiacs.umd.edu/~lsd/papers/PLS-ICCV09.pdf. 2009.

[41] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. http://arxiv.org/pdf/1409.1556.pdf. 2014.

[42] Paul Viola and Michael J. Jones. *Robust Real-Time Face Detection*. http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf. 2003.

[43] S. Walk et al. "New features and insights for pedestrian detection". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. 2010.

[44] Liming Wang et al. *Penn-Fudan Database for Pedestrian Detection and Segmentation*. https://www.cis.upenn.edu/~jshi/ped_html/. 2007.

[45] Xiaoyu Wang, Tony X. Han, and Shuicheng Yan. *An HOG-LBP human detector with partial occlusion handling*. http://web.missouri.edu/~hantx/paper/Wang_Han_Yan_iccv09.pdf. 2009.

[46] C. Wojek, S. Walk, and B. Schiele. "Multi-cue onboard pedestrian detection". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. 2009, pp. 794–801. DOI: 10.1109/CVPR.2009.5206638.

[47] Christian Wojek and Bernt Schiele. *A Performance Evaluation of Single and Multi-feature People Detection*. https://www.d2.mpi-inf.mpg.de/node/268. 2008.

[48] Bo Wu and R. Nevatia. "Detection of multiple, partially occluded humans in a single image by Bayesian combination of edgelet part detectors". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 1. 2005, 90–97 Vol. 1. DOI: 10.1109/ICCV.2005.74.

[49] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. "Online Object Tracking: A Benchmark". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013.

[50] Z. Wu et al. "Early Hierarchical Contexts Learned by Convolutional Networks for Image Segmentation". In: *Pattern Recognition (ICPR), 2014 22nd International Conference on*. 2014, pp. 1538–1543. DOI: 10.1109/ICPR.2014.273.

[51] C. T. Zahn. "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters". In: *IEEE Transactions on Computers (Volume:C-20 , Issue: 1)*. 1971.

[52] Matthew D. Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks*. https://www.cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf. 2009.

[53] Matthew D. Zeiler et al. *Deconvolutional Networks*. http://cs.nyu.edu/~fergus/drafts/utexas2.pdf. 2010.