UPPSALA
UNIVERSITET

Examensarbete 30 hp
Juni 2018

Study and Analysis of Convolutional Neural Networks for Pedestrian Detection in Autonomous Vehicles

Louise Augustsson

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Study and Analysis of Convolutional Neural Networks for Pedestrian Detection in Autonomous Vehicles

Louise Augustsson

The automotive industry is heading towards more automation. This puts high demands on many systems like Pedestrian Detection Systems. Such systems need to operate in real time with high accuracy and in embedded systems with limited power, memory resources and compute power. This in turn puts high demands on model size and model design. Lately Convolutional Neural Networks (ConvNets) have dominated the field of object detection and therefore it is reasonable to believe that they are suited for pedestrian detection as well. Therefore, this thesis investigates how ConvNets have been used for pedestrian detection and how such solutions can be implemented in embedded systems on FPGAs (Field Programmable Gate Arrays). The conclusions drawn are that ConvNets indeed perform well on pedestrian detection in terms of accuracy but to a cost of large model sizes and heavy computations. This thesis also comes up with a design proposal of a ConvNet for pedestrian detection with the implementation in an embedded system in mind. The proposed network performs well on pedestrian classification and the performance looks promising for detection as well but further development is required.

Handledare: Detlef Scholle
Ämnesgranskare: Carolina Wählby
Examinator: Tomas Nyberg
ISSN: 1401-5757, UPTEC F** ***

Populärvetenskaplig sammanfattning

I dagsläget går fordonsindustrin mot allt mer automatisering. I viss utsträckning finns det redan helt autonoma fordon och delvis autonoma fordon är inte längre något extraordinärt på grund av det succesiva införandet av avancerade förarassistanssystem (*Advanced Driver Assistance Systems*, förkortat *ADAS*). ADAS är system som aktivt assisterar föraren för att förhindra olyckor och öka komforten i körningen. Exempel på sådana system kan vara system som varnar föraren om fordonet åker utanför väglinjerna, system som hjälper föraren med parkering och system som detekterar, varnar för och till och med bromsar in för fotgängare. Just detektering av fotgängare är ett nödvändigt steg på vägen mot fullt autonoma fordon. Av uppenbara skäl måste ett sådant system verka i realtid och ha mycket hög noggrannhet. Det ställer stora krav på systemets funktionalitet. Det måste dessutom vara integrerat i ett inbyggt system med begränsade minnes-, energi- och beräkningsresurser.

Senare år har artificiella faltnings-baserade neurala nätverk (*convolutional neural networks*, förkortat *ConvNets*) revolutionerat områden så som bildklassificering och objektdetektering. Därför är det ett rimligt antagande att sådana nätverk även skulle vara lämpade för att detektera fotgängare. Fördelarna är att nätverken lär sig extrahera egenskaper i bilden som sedan kan fogas samman till en slutsats om vad bilden innehåller. Detta genom att träna på ett stort antal exempelbilder där innehållet är känt. På så vis behövs ingen handskriven specifikation för vilka bildegenskaper som definierar en fotgängare. En sådan specifikation kan nämligen vara väldigt svår att skriva eftersom fotgängare uppenbarar sig i så många olika kroppsposeringar, i olika miljöer, med olika kläder, i olika ljusförhållanden med mera. Nackdelarna är att nätverksmodellerna ofta är väldigt stora och beräkningstunga, vilket inte är fördelaktigt för implementering i ett inbyggt system.

Enkelt beskrivet består beräkningarna i ett ConvNet till största del av en stor mängd flyttals multiplikationer. Dessa kan i viss mån göras parallellt, det vill säga att flera beräkningar genomförs samtidigt. Till dessa parallellberäkningar används ofta datorns grafikkort (*graphics processing unit*, förkortat *GPU*). GPUer är inte alltid optimala att ha i inbyggda system bland annat på grund av deras relativt höga energikonsumtion. Ett alternativ är så kallade 'på-plats-programmerbara grindmatriser' (*field programmable gate arrays*, förkortat *FPGA*). FPGAer har fördelarna att vara förhållandevis energisnåla, snabba och optimerade för parallellberäkningar. Däremot har de begränsat minne och är inte optimala för flyttalsoperationer. FPGAer är därför lovande för att accelerera neurala nätverk men det finns vissa ut-

maningar som först behöver övervinnas.

Det här arbetet syftar till att undersöka hur ConvNets idag används för detektering av fotgängare och vilken potential det finns att implementera sådana nätverk på FPGAer. De dragna slutsatserna är att det finns mycket forskning gjort på området men att nätverken som används oftast är beräkningstunga och optimerade för GPUer och inte för inbyggda system. Arkitekturen består oftast av tre steg där det första steget föreslår områden i bilden som potentiellt innehåller en fotgängare, det andra steget extraherar egenskaper hos det föreslagna området och det sista steget klassificerar om området innehåller en fotgängare eller inte. Arkitekturen uppnår goda resultat i termer av noggrannhet men inte i termer av snabbhet eller i antal beräkningar. Det finns därför ett behov av att utveckla nätverk som är bättre optimerade för inbyggda system och FPGAer. Detta kan uppnås genom smart design med mindre modellstorlekar som kräver mindre antal beräkningar. Metoder så som beskärning och kvantisering är lovande för att möjliggöra implementering på FPGAer. Beskärning innebär att de delar av nätverket som har en liten påverkan på resultatet bokstavligen skärs bort och kvantsering betyder att nätverksparametrarna omvandlas från flyttal till fixa tal.

Vidare syftar arbetet till att hitta en nätverksdesign för detektering av fotgängare med möjligheten för implementation i ett inbyggt system i åtanke. Den föreslagna designen bygger på ett nätverk som är förtränat på en stor mängd bilder av olika slag plus ett lager som extraherar en sannolikhetskarta för var det finns fotgängare i bilden. På så vis behövs inte det första steget i den traditionella arkitekturen som är beskriven ovan och antalet beräkningar minskar. Det visade sig att det förtränade nätverket efter ytterligare träning presterar bra på att klassificera om en bild innehåller en fotgängare eller inte samt att sannolikhetskartan fungerar för enklare problem. Dock uppnåddes inga starka resultat för detektering av fotgängare. Istället föreslås nödvändig vidareutveckling för att få detekteringen att fungera och framför allt för att fungera i ett inbyggt system.

Contents

List of Figures

List of Tables

Acronyms

ACC Adaptive Cruise Control.

ACF Aggregated Channel Features.

ADAS Advanced Driver Assistance System.

AMASS Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems.

AMI Amazon Machine Image.

AOs Amount of Operations.

AWS Amazon Web Services.

CACC Cooperative Adaptive Cruise Control.

Caffe Convolution Architecture For Feature Extraction.

ConvNet Convolutional Neural Network.

CPU Central Processing Unit.

cuDNN CUDA Deep Neural Network library.

EC2 Elastic Compute Cloud.

FPGA Field-Programmable Gate Array.

GPU Graphical Processing Unit.

HDF Hierarchical Data Format.

HDL Hardware Description language.

HOG Histograms of Oriented Gradients.

ILSVRC ImageNet Large-Scale Visual Recognition Challenge.

IP Intellectual Property.

LDCF Locally Decorrelated Channel Features.

LMDB Lightning Memory-Mapped Database.

PPS Pedestrian Protection system.

R-CNN Region-based Convolutional Neural Networks.

ReLU Rectified Linear Unit.

RPN Region Proposal Network.

SafeCOP Safe Cooperating Cyber-Physical Systems using Wireless Communication.

SGD Stochastic Gradient Descent.

SSH Secure Shell.

Chapter 1

Introduction

The automotive industry is heading towards more automation. To some extent there already exist fully autonomous vehicles and semi-autonomous vehicles are no longer extra ordinary due to the entrance of Advanced Driver Assistance System (ADAS). ADAS are systems designed to actively assist the driver and several leading car and truck manufactures have implemented a lot of ADAS in their vehicles. Examples of such systems are Pedestrian Protection system (PPS), lane departure warning systems and blind spot monitors [1]. This automation of course implies great challenges in the technology development but also in other fields. For example, the ethical aspects must be taken into consideration and ethical dilemmas do not always have simple solutions.

This chapter introduce the reader to the subject of this project. It starts with Background (section 1.1) and Motivation (section 1.2). Then it continuous with a description of the Purpose (section 1.3) and Goals (section 1.4) and finishes with Ethical considerations (section 1.5) and Delimitations (section 1.6).

1.1 Background

To drive a car, the driver must accomplish several complicated tasks simultaneously. The driver must, except from the obvious task to gas, steer and brake, also be able to interpret many, mainly visual, stimuli and know how to react on them. Examples of this are identifying the road and know where on it to drive, detecting traffic signs and know their meaning, identifying possible dangerous situations and detecting objects such as other vehicles and pedestrians. A human driver can accomplish this without greater effort. The scenes in the environment are analysed and processed subconsciously. Since the human driver has learned what the traffic signs look like, their meaning and how to react on them, he/she knows what to do when they show up besides the road. A human driver can recognize a pedestrian by seeing a leg since he/she understands that a leg must belong to a body and he/she is able to predict what will happen in certain situations because it is learned from previous, similar situations.

All the above described, and required for driving, skills are not as simple for a computer as for a human [2]. The challenge for computers to understand images lays in the field of computer vision where deep learning and neural networks recent years have performed state of the art [3],[4],[5],[6],[7].

1.1.1 Convolutional Neural Networks

Hubel and Wiesel [8] showed in 1962 that certain neural cells in the brain respond to different visual stimuli, such as edges of different directions. The neural cells are connected to each other to form layers which in turn form a network. Within tens of a millisecond, the network in the visual cortex is able to analyse a complex scene and classify a large number of objects.

The main idea with artificial neural networks is to mimic the visual cortex of a human or an animal. An artificial network is built up of layers of neurons with learnable parameters, which are updated during training [9]. Different neurons in the network identify different features, just like the non-artificial neural cells. The networks can be trained for a number of different tasks, e.g. image classification and object detection.

A Convolutional Neural Network (ConvNet) is a type of a neural network. The most important layers in a ConvNet are the convolutional layers, in which the neurons convolve over the input image in order to be able to detect the same features at different positions in the image. The reason for this and the main idea idea behind ConvNets is that an object is the same, regardless of where in the image it is located [10].

1.1.2 AMASS and SafeCOP

The project is conducted at Alten which is partner in two ongoing European projects; AMASS[1] [11] and SafeCOP [2] [12]. AMASS is creating an European-wide open tool platform for assurance and certification and has the goal to lower certification costs for cyber-physical systems. The AMASS project will be benchmarked by a number of industrial case studies where case study 3 is of interest for Alten. Case study 3 is about collaborative automated fleets of vehicles.

SafeCOP targets cyber-physical systems-of-systems whose safe cooperation relies on wireless communication. Within SafeCOP there are five different use cases that target different aspects where wireless communication can be of use. For Alten use case 3 is of relevance. It is concluded as vehicle control loss warning and includes auto-braking and platooning.

The two project have the common goal to promote cooperation between European companies so that information and knowledge can be shared in order to strengthen the European position in technology development.

1.1.3 Altens' physical demonstrator

Two radio controlled vehicles in scale 1:8 relative to real vehicles are forming a physical demonstrator for systems developed by Alten and Altens' project partners. Previous works on the vehicles have inter alia been about line following, Adaptive Cruise Control (ACC), data aggregation and criticality systems. At the start of this project the vehicles had an implemented mixed criticality system, capable of following clear lines on the road and drive in a platoon using ACC.

The goal is to implement the work done during this project on the platform on the demonstrator. The platform consists of a Z-turn board with zynq -7020 [13] and the the implementation will be on the programmable logic of the board (on the Field-Programmable Gate Array (FPGA)).

1.1.4 Related work

This project is a part of a larger project at Alten, carried out by five master students. The focuses of the other works can roughly be divided into four areas: implementation of a controller for Cooperative Adaptive Cruise Control (CACC), implementation of a trajectory planning algorithm for overtaking scenarios, implementation of an active safety monitoring system and implementation of a traffic sign detection system. The last project is most related to this project. Different traffic sign detection algorithms are studied during the literature study, including ConvNets, and one of the algorithms, called template method, is chosen to be implemented on the FPGAs on the physical demonstrator. Unlike this project the traffic sign detection project mainly focus on the hardware rather than on the algorithms them self. The

[1] AMASS - Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems

[2] SefeCOP - Safe Cooperating Cyber-Physical Systems using Wireless Communication.

project investigates how the hardware can be used to accelerate the algorithm. This will be of importance for future development of this project as well. The other three projects are not directly related to this project more than that they also treat the field of autonomous vehicles and that they together with this project contribute to the development of the same autonomous vehicles.

1.2 Motivation

In urban environments pedestrians and vehicles share the same space and move closely to each other. This puts high demands on PPS in autonomous vehicles to prevent collisions and accidents. Challenges for such systems are that they must be safety critical since a failure or design error has a potential to cause loss of life [14], they need to function in real time and have high accuracy. Furthermore it needs to be possible to implement them on devices with limited compute power to operate in real time.

Challenges with the detection mainly lay in the many appearances of pedestrians. They appear in a wide variety of articulated poses, with variable clothing and at different distances to the camera. The background is often complex and the illumination can differ a lot. This makes it hard to specify features that cover all possible appearances.

In this project a ConvNet will be used instead of doing the feature specification manually as would be necessary with many other object detection methods. ConvNet have recent years achieved state of the art performances on a variety of problems in the field of image classification and detection [3],[5]. This makes it reasonable to believe that they at least are good candidates for real world applications in autonomous vehicles. Furthermore, a ConvNet can be re-trained if the environment in which the vehicle operates changes drastically, e.g. if the background in the images changes or if the human movement pattern or appearance change due to new technology or other circumstances. If this would be the case, new features do not need to be specified, instead the network can be re-trained on a new set of images. It is also possible to re-train a network to detect other objects than pedestrians, if desired.

1.3 Purpose

Pedestrian detection is a necessary component for autonomous vehicles. So to put this project in a broader context: it aims to be one of the required steps toward fully autonomous vehicles that can operate in environments shared with unprotected pedestrians without running unnecessary high risks. As said in section 1.1.2, this project is related to the two European projects, AMASS and SafeCOP, that aim for increased cooperation and information sharing between the participating companies. One of the purposes with this project is hence to contribute to the European projects. Since platooning is a case for both projects, it is reasonable to see that further

development of this project could include that the vehicles inform each other if they detect pedestrians that the other vehicles have not yet detected.

1.4 Goals

Since this project is a part of a larger project carried out by master students with different backgrounds and different viewpoints, there are both individual goals and team goals. The individual goals are a subset of the team goals. In the individual project ConvNets for pedestrian detection will be studied and analysed. The goal is to design a system for pedestrian detection which is possible to implement in an embedded system and fast enough to operate in real time.

The team goal is to develop a demonstrator consisting of two vehicles that can safely handle a use case. The demonstrator consists of two vehicles whereof one follows the other using CACC. The use case consists of the following traffic situations.

1. The first vehicle detects a traffic sign telling it to slow down.
2. The first vehicle slows down and alerts the second vehicle about the speed-change.
3. The first vehicle detects a pedestrian on the road.
4. The first vehicle stops completely to avoid a collision with the pedestrian.
5. The second vehicle receives information from the first vehicle when it starts braking.
6. Finally, the second vehicle plans for an overtake and performs it if possible.

1.5 Ethical considerations

In the field of autonomous vehicles the ethical aspect must always be taken into consideration. Even if the automation has potential to decrease traffic accidents [15] there will always be situations involving unavoidable harm that requires the vehicle to make difficult ethical decisions. One thought experiment is the famous Trolley problem [16]. A trolley is heading on a track and if nothing is done it will kill five persons. You have the option to steer it to another track, killing one person instead. What would you do? Figure 1.1 visualises three twists of the Trolley problem, modernised for the era of autonomous vehicles. The vehicle has to choose between killing one or several pedestrians or its own passengers. What should it be programmed to do? This kind of questions cause a clear division in peoples' opinions. In a study carried out by Bonnefon[17], a majority of the participants wanted other people to buy a vehicle that sacrifices its own passengers for the greater good, but would rather want to buy a car that did the opposite. Even if this kind of situations seem to be very unlikely to occur, they need to be though of and prepared for to enable fully autonomous vehicles.

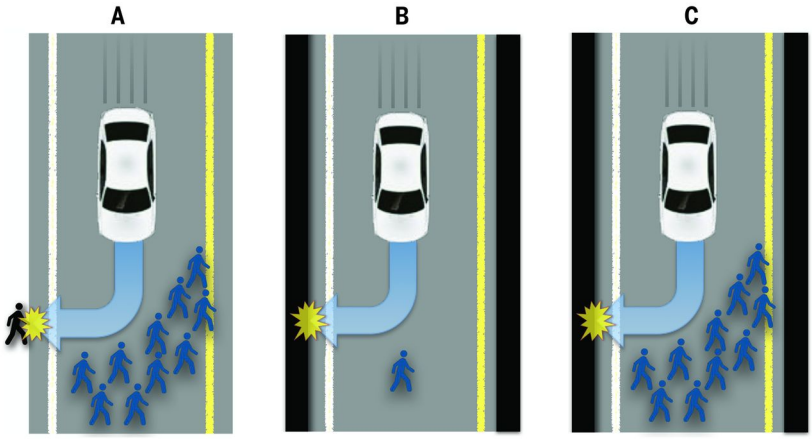


Figure 1.1: Modern versions of the Trolley problem. The vehicle must decide between (A) killing several pedestrians or one passerby, (B) killing one pedestrian or its own passenger, and (C) killing several pedestrians or its own passenger [17].

In the meantime this project was carried out, the first fatal accident involving an autonomous vehicle and a pedestrian was reported [18]. An autonomous vehicle collided with a pedestrian that was walking on the street, outside of a pedestrian crossing. Yet, all circumstances are not clear and it is hard to determine whose fault it was. But still, accidents like this of course arises questions about safety, security and responsibility. How can the vehicles be guaranteed to be as safe as possible and how will the general public be convinced that this is the case? Without the general public's acceptance it does not matter how safe the vehicles are, they will not be allowed to traffic the streets anyway. When it comes to responsibility there is a need to have clear laws of whom is responsible. Other questions concern security. How can it be assured that the vehicles are secure enough? It is not an easy task to assure that all systems function and that it is close to impossible for an unauthorized to access and take over the systems.

When it comes to the specific topic of detecting pedestrians, or humans, it is important to have in mind what it can be used for. The field of autonomous vehicles can maybe be seen as something "good" and other fields like camera surveillance or robotics as well in some senses. But human detection can also be of interest when it comes to the development of advanced weapon system and other types of ethical questionable areas.

1.6 Delimitations

This project will be limited to only cover the detection of pedestrians. It will not classify whether the pedestrian is an adult or child, if he/she is on or besides the road or in which direction he/she moves. The project will not include what action the vehicle should take in certain situations and it will not include any communication between the vehicles. The reader should be aware of that there exist many other methods than ConvNets suited for the task at hand even though this project is limited to only cover the ConvNet approach.

The project is limited to only last during 20 weeks inclusive the writing of an academic report, a presentation and opposition on another thesis work. The project corresponds to 30 credits and is done within the frames of the master programme in engineering physics at Uppsala University.

1.7 Project outline

This report is divided into four parts. The first part is a pre-study which results in the Introduction chapter (chapter 1) and the Methodology chapter (chapter 2). The time duration of the first part is approximately two weeks. The second part is a literature study with time duration 8 weeks. The literature study covers the *theory behind ConvNets*, *ConvNets for pedestrian detection* and *ConvNets and FPGAs* which is presented in chapters 3, 4 and 5. The third part covers the implementation which also has a time duration of 8 weeks. Chapter 6 presents the datasets that are used in the project, chapter 7 describes the design and implementation and chapter 8 presents the results from the work.

The two last weeks of the project are dedicated to summarising and concluding the project as well as finalising the report, present the project and oppose on another thesis. Conclusions drawn from this project and future work are presented in chapter 10.

Chapter 2

Methodology

This chapter presents the methodology used in the literature study and the design and implementation phase. During the literature study relevant research are studied with purpose to increase the understanding of what has already been done and to make it possible to in some extent reuse or redesign already existing methods in the design and implementation part. The literature study leads to chapters 3, 4 and 5 in the report which cover relevant theory and related work. During the second phase of the project a ConvNet for pedestrian detection is designed and evaluated.

2.1 Literature study

The literature study is carried out in a way proposed by Creswell [19]. The proposed way consists of six steps which are described below.

1. Identify keywords relevant to the project. The keywords will be used when searching for material to be studied. Keywords relevant for this project are
   - Deep learning, neural network, convolutional neural network
   - Object detection, pedestrian detection
   - Image dataset, annotated image dataset, pedestrian dataset
   - FPGA, FPGA with deep learning, FPGA with deep learning for pedestrian detection
2. Search for relevant material, having the keywords in mind. In this project mainly databases available online are used. The used databases are
   - ACM digital library (Association for Computing Machinery)
   - arXiv
   - IEEE Explore Digital Library
   - Science Citation Index Expanded
   - ScienceDirect
3. Locate articles and books relevant to the project. Skim through the articles and chapters and set a priority on them, depending how important they seem to be for the project and whether they will contribute to the understanding of the literature or not.
4. Design a literature map. A literature map is a visual representation over how the studied literature is connected and how this project will contribute to the existing research. This map is only a tool for increased understanding of the theory and will not be shown in the report.
5. Draft summaries of the most relevant articles.
6. From the summaries, conclude what information is most relevant for the project and what should be written in the report. Write relevant theory and related work which will constitute the literature study part of the report.

It is worth mentioning that the literature study is of qualitative manner which means that no quantitative results are presented. The goal with the literature study is to place the benchmark in the field and determine which pieces are missing. A well performed literature study enables the contribution of the project to be valuable for the field.

2.2 Design and implementation

The second part is of quantitative manner and the results are presented in a chapter 8, Results. The initial goal with this part was to design a ConvNet and implement it on the physical demonstrator. Due to time limitations the implementation is not carried out. The ConvNet is only designed with a future implementation in mind. This part of the project is conducted by following the below presented steps where only step 1-4 is carried out.

1. From the literature study, identify missing pieces of work.
2. Set up requirements that the design should fulfil.
3. Based on the gained knowledge from the literature study and with the requirements in mind, set up a preliminary design
4. Evaluate the design and find possible improvements.
5. Implement the designed network.

The design and implementation phase is an iterative process. Especially the third and fourth steps are conducted together in order to reach as high results as possible.

Chapter 3

Convolutional Neural Networks

This chapter goes through the theory behind ConvNets needed in this project. It starts with a brief presentation of the history of ConvNets (section 3.1) and continues with presenting the different components of a ConvNet such as the different layers, training, dropout and initialisation of weights (section 3.2). Finally it presents three network architectures that are of importance in this project (section 3.3). Below follows a list of variables that are used in this chapter.

AOs amount of operations

b bias

η learning rate

k filter size

K filter

λ constant

m number of output feature maps

n number of input feature maps

R_o size of output feature map

W weight matrix

x weight

X input feature map

Y output feature map

3.1 Brief history

ConvNets are not as new as most people seems to believe, in fact they have been around since the 80's. In 1980 Fukushima introduced Neocognitron [20] which was a convolutional-like network that served as inspiration for the network architecture that is seen as ConvNets today. During the 80's Yann LeCun first started the work that led to LeNet5 [10] which is often referred to as the first ConvNet. In the beginning the network was used for character recognition and it has been an inspiration for many of today's network architectures. Due to the lack of both powerful hardware and large dataset the development of ConvNets went very slowly the following years after LeNet5. Time passed by and more and more data became available and the hardware became more and more powerful, which in combination with more efficient training algorithms paved the way for the resurrection of ConvNets.

In 2012 a ConvNet called AlexNet [3] won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [21] with a top 5 error rate of 15.4% improving the earlier top result of 26.2%. Ever since, different ConvNets have continued reducing the error rates in the ILSVRC by record levels [4],[5],[6],[7]. Today ConvNets are state of the art and even perform better than humans when it comes to image classification, object detection and a number of other fields.

3.2 Network components

A ConvNet consists of neurons connected to each other in such a manner that they form layers in a network. The first layer is called the input layer, the final layer is called the output layer and the intermediate layers are called hidden layers. The input to each layer is called an input feature map and the output is called an output feature map. The input feature map to the first layer is an image, which is a set of arrays, and the input/output feature maps between the layers are abstract arrays of pixels. The different types of layers in a ConvNet are usually convolutional layers, pooling layers and non-linearity layers, described in section 3.2.1. In order get satisfying predictions the network must be trained on a large amount of annotated data, e.g. data where the ground truth, or correct label, is given. If the amount of training data is not enough or the number of network parameters is too large, there is a risk of overfitting. Overfitting means that the network adapts to the training data and achieve strong results on the training data but underperforms on new, unseen data such as the validation data. There is a number of ways to avoid overfitting where introducing dropout during training is one. Overfitting and dropout are described in section 3.2.3. The training consists of iteratively forward and backward passing the training data while updating the network weights, described in section 3.2.2. How these weights are initialised have a strong impact on the network performance. Different ways of initialising the network weights are described in section 3.2.4.

3.2.1 Network layers

Convolutional layer The convolutional layers are, as the name suggests, the foundation of ConvNets. The word *convolutional* refer to that the layers consist of filters that *convolve* with the input feature maps. The filters are built up of weights that are piecewise multiplied with the input. The output is given by the multiplication plus a bias value. Mathematically, the output from a convolutional layer is given by

$$Y_i = b + \sum_{j=1}^{n} X_j \otimes K_{ij} \quad (i = 1, 2, ..., m) \tag{3.1}$$

where Y is the output feature maps, m is the number of output feature maps, n is the number of input feature maps, X is the input feature maps, K the filter and $\otimes$ represents the convolution [22]. The Amount of Operations (AOs) in a convolutional layer is given by

$$\text{AOs} = k \times k \times n \times m \times R_o \times R_o \times 2 \tag{3.2}$$

where k is the size of the filters if all filters are assumed to have the same size and R_o is the size of the output feature maps. Figure 3.1 is an illustration of a convolutional layer where a filter convolves with an input feature map and produces an output feature map. Here $n = 1$, i.e. there is only one input feature map. The blue field on the input feature map is called the filters' receptive field and can be thought of as what the filter sees at the moment. Each filter represents a certain feature and if that

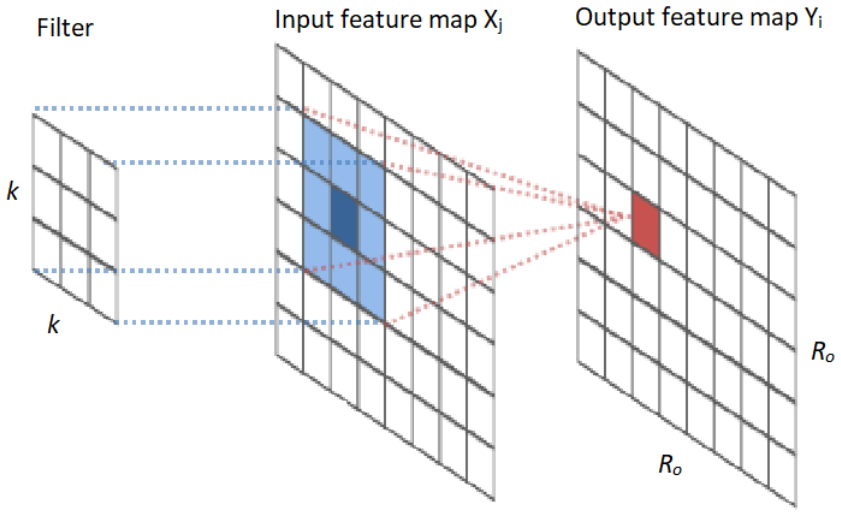


Figure 3.1: A filter is convolved with the input feature map to produce an output feature map.

certain feature exists in the input feature map, the piecewise multiplication results in a larger output. Convolving the filter with the input feature map means that the same feature can be found at different positions in the input. The first convolutional layers extract lower level features and when several convolutional layers are cascaded they can extract higher level features as well. A convolutional layer have size, stride

and padding. The size corresponds to the filter size, stride is how far the filter is moved in each step and padding is used to control the size of the output. Zero padding of size 1 means that the input is padded with one layer of zeros around the boarder.

Pooling layer In the pooling layers, also referred to as down sampling layers, the spatial size of the input is reduced. There exist different types of pooling layers where average pooling and max pooling are the most commonly used. A pooling window is convolved with the input and for each subregion it outputs the average or maximum value. Figure 3.2 is an illustration of max pooling where each colour represents a pooling window. Down sampling fulfils several purposes. Firstly it decreases the number of parameters, which in turn both decreases the computation cost and prevents overfitting [23]. Using a pooling window of size 2x2 and stride 2 decreases the input size with 1/4, for example. Secondly the convolutional layers after a pooling layer sees a larger part of the initial image which makes it possible to extract higher level features. The idea is that once the lower level features are extracted their exact positions are not as important as their relative positions. It is most common to use non-overlapping pooling which is when the stride equals the size of the pooling window but overlapping pooling is an alternative, where the stride is less than the size of the pooling window. The AOs in a pooling layer is given by

$$\text{AOs} = m \times R_o \times R_o \times \lambda \tag{3.3}$$

where λ is a constant depending on pooling method and size of pooling window [22].

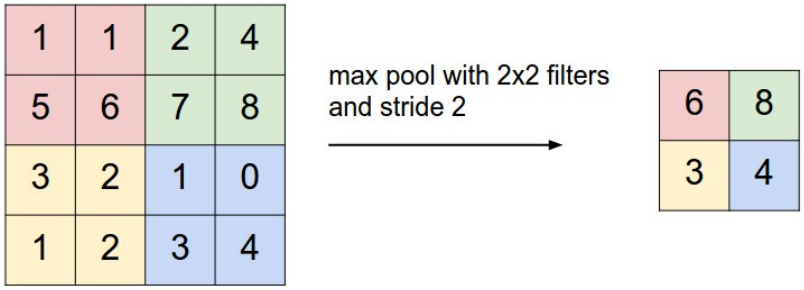


Figure 3.2: Max pooling to down sample input feature map [23].

Non-linearity layers Since real world data is rarely or never linear, it is not suitable to use a linear model to classify real world data. Therefore, non-linearity must be inserted into the models and non-linearity layers are used in this purpose. In these layers, an activation function operates pixel wise on the input and returns the function value for that pixel value. tanh, sigmoid and Rectified Linear Unit (ReLU) are commonly used activation functions. They are all illustrated in Figure 3.3. As can be seen, the input must be normalised if the sigmoid or tanh activations are used. ReLUs are commonly used in larger networks since they require less computational power [3] and prevent saturation [23]. The ReLUs activation function is given by

$$Y_{i,j} = \max(0, X_{i,j}) \tag{3.4}$$

and the AOs needed is given by

$$AOs = m \times R_o \times R_o \tag{3.5}$$

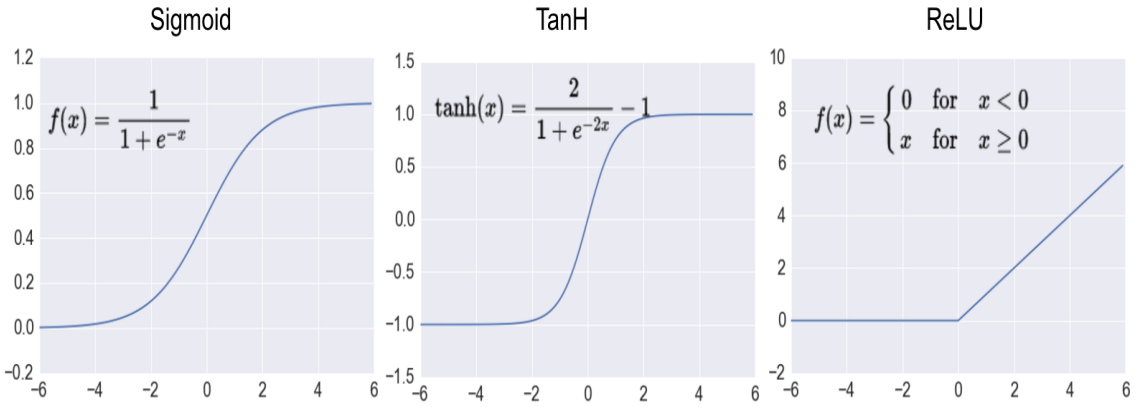


Figure 3.3: The sigmoid, tanh and ReLU activation functions.

Fully connected layers Basically a fully connected layer is a convolutional layer where all neurons are connected to all neurons in the adjacent layer. The weight matrix, W, reflect the strength of each joint [22]. Mathematically a fully connected layer can be represented by

$$Y_i = \sum_{j=1}^{n} X_j * W_{i,j}. \tag{3.6}$$

Fully connected layers learn non-linear combinations of the features extracted in the convolutional layers. For image classification they are always last in a ConvNet since the output is a 1D vector which cannot serve as input to a convolutional layer. In the case of object classification the last fully connected layer must always consist of as many neurons as there are classes. The AOs needed in a fully connected layer is given by

$$AOs = m \times n \times 2 \tag{3.7}$$

Softmax The final layer in a ConvNet is always some kind of classifier that classifies the features extracted in the earlier layers. For this a soft max layer is commonly used. Given the weights from the last fully connected layer the softmax function calculates the probability that the input belongs to a certain class. The soft max function is given by

$$p_j = \frac{e^{X_j}}{\sum_j^n e^{X_j}} \tag{3.8}$$

where p is the probability that the object is in class j and n is the total number of classes.

3.2.2 Network training

Training is used in order to teach the network to make correct predictions with high accuracy. There are two ways in which training can be conducted: supervised or unsupervised training. Supervised means that the ground truth is known and given during the training process and unsupervised means that it is not. Unsupervised training can hence not be used for tasks like classification and detection but it can be used for extracting hidden structures in the data. There are several different approaches for unsupervised training of ConvNets but they all have in common that the input is clustered depending on similarities in the data. The supervised training process can be described in three steps. First, the input data is passed forward through the network and a prediction is produced at the output layer. Then the prediction is compared to ground truth and a loss function is computed. The loss function is a measure of accuracy within machine learning. Finally the network weights are updated according to how much impact they had on the loss function and according to a learning rate. The learning rate is a parameter that determines how large impact the gradient of the loss function has on the weight update. The second and third steps are called back propagation [24]. The goal with the weight updating is to minimize the loss function. This is done by first computing the gradient of the loss with respect to each of the weights, using the chain rule. Then each weight is updated according to for example Stochastic Gradient Descent (SGD), given by

$$w_{new} = w_{old} - \eta \frac{\partial E}{\partial w_{old}} \tag{3.9}$$

where $w_{new/old}$ is the new/old weight value, η is the learning rate and $\frac{\partial E}{\partial w_{old}}$ is the gradient of the loss function with respect to the old weight value.

3.2.3 Dropout

As overfitting is undesired and it is not always possible to avoid it by increasing the amount of training data or decreasing the number of network parameters, other methods have to be applied. One commonly used is dropout [25] [26] which is a regularisation method where randomly selected neurons are ignored during training both in the forward pass and backward propagation. This means that the selected neurons neither affect the prediction nor are updated in the backward propagation. The effect of dropout is that the network becomes less dependent on single neurons and can generalise better. Worth noting is that dropout is only used during training since it decreases the prediction results.

3.2.4 Initialising of weights

When designing a network from scratch it can be a hard task to initialise the weights in a good way. They can neither have too small values, have too large values nor be constant. The most naive way would be to initialise all weights to zero, but that has shown to have negative consequences. If all weights share the same value, the gradient of the loss function with respect to the weights become the same and hence

the weights are updated equally. This implies a symmetric network which is not desirable. Instead the weights are usually initialised randomly from a Gaussian or Normal distribution.

Another way in which the weights can be initialized is to use the weights from a pre-trained model. This is called transfer learning. The model is commonly trained on the ImageNet dataset due to the large amount of data and large amount of object classes. There are three methods in which the network can be adapted to the task at hand.

Fixed feature extractor New layers are added to the pre-trained network with weights initialised as described above. The new network is trained while the ordinary weights are kept constant and the new are updated. This is suitable when the task which the network is pre-trained for and the task at hand are very similar. Then the same features extracted by the pre-trained network are used in the new network.

Re-training The last layer from the pre-trained network is replaced with a customized layer and the whole new network is trained with a decreased learning rate. This is suitable when there is a larger difference between the task which the network is pre-trained for and the task at hand.

Combination A combination of the above described methods can be used. For example new network layers are added, some of the pre-trained network weights are kept constant and the rest of the weights are updated during training. The first layers in a network extract lower level features such as edges, corners and colours. Since these features are the same for most objects that part of the pre-trained network can remain unmodified and function as a generic feature extractor applicable for various tasks.

3.3 Network architectures

With the start of AlexNet in 2012 the focus on increasing accuracy dominated the development of ConvNets for image classification. Deeper and deeper networks were introduced, improving the accuracy but to the cost of an increased number of parameters and an increased need of compute power. At the meantime, work was done for decreasing the number of network parameters while keeping the high accuracy. In 2016 Iandola *et al.* [27] presented a network with 50 times fewer parameters than AlexNet but with the same accuracy, SqueezeNet. Later Verbickas [28] adapted SquuezeNet for pedestrian detection in a network named SqueezeMap.

3.3.1 AlexNet

At the time AlexNet was introduced, it was one of the largest existing ConvNets and it achieved the best result ever reported on the ILSVRC. The network is built

up of five convolutional layers followed by three fully connected layers and a softmax with overlapping max pooling and ReLUs in between. It was shown that using overlapping pooling instead of non-overlapping pooling decreased the risk of overfitting and using ReLU instead of sigmoid or tanh decreased the computational cost while not decreasing accuracy. The convolutional layers consist of filters of different sizes, 3x3, 5x5 and 11x11. These layers in combination with the fully connected layers takes more than 99% of the processing time. Figure 3.4 shows the network architecture from the original paper [3] where the two pipelines represents the optimized GPU implementation. Half the network weights are saved on one of two GPUs and the rest on the other GPU. Communication between the two GPUs only occurs at certain layers, marked with dotted lines in the figure.

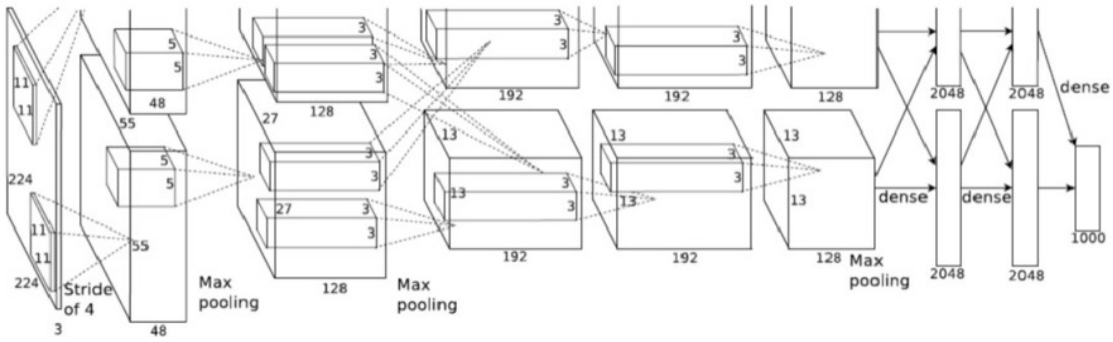


Figure 3.4: AlexNet network architecture [3].

3.3.2 SqueezeNet

The primary goal with SqueezeNet was to build a ConvNet with minimised number of parameters but still achieving state of the art accuracy. SqueezeNet consists of 50 times fewer parameters than the AlexNet-architecture and achieves the same accuracy. Three strategies was used when designing the network. Firstly, 1x1 filters were used instead of 3x3 filters since a 3x3 filter has nine times more parameters than a 1x1 filter. Secondly the number of input channels to the 3x3 filters were decreased, also in order to minimize the number of parameters. Thirdly, the network was down sampled at a late stage, since it was shown to maximize the accuracy. The network was realised by the introduction of so called fire modules. A fire module consists of two layers, a squeeze convolution layer with only 1x1 convolution filters followed by an expanding layer consisting of both 1x1 and 3x3 convolution filters [27]. The final network architecture from the original paper is shown in Figure 3.5. The network begins with a convolutional layer followed by nine fire modules, another convolutional layer and a softmax. After each convolutional layer, ReLUs were used as activation function and max pooling was inserted after fire module 4 and 8 and after convolutional layer 1. Between convolutional layer 10 and the softmax global average pooling is used.

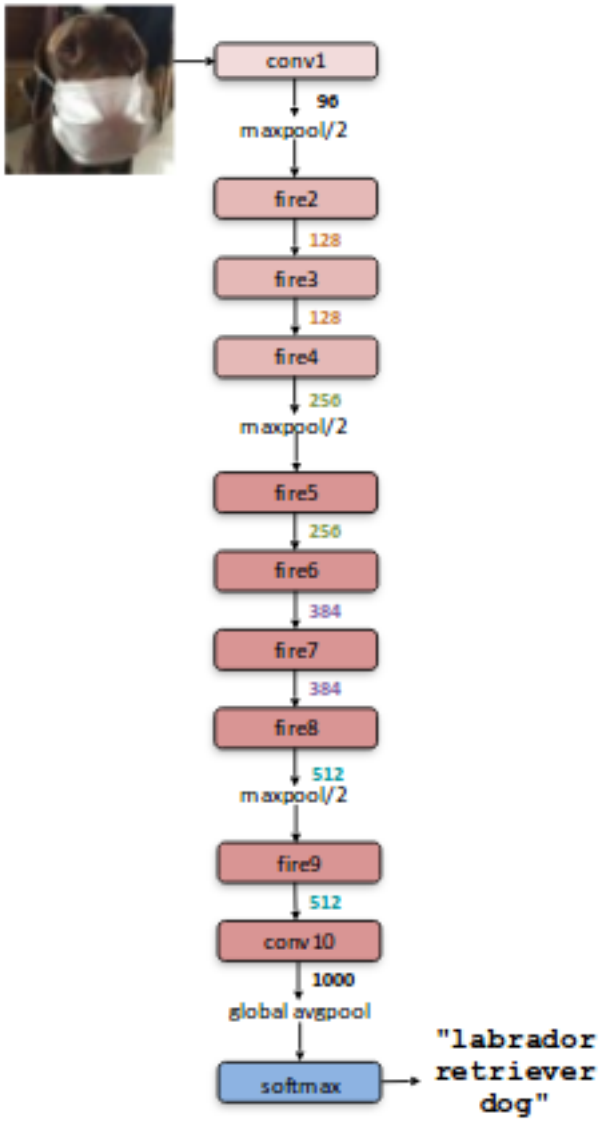


Figure 3.5: SqueezeNet network architecture [27].

3.3.3 SqueezeMap

The SqueezeMap network architecture is based upon the SquuezeNet architecture and is adapted for pedestrian detection. SqueezeMap only differs slightly from SqueezeNet in the hidden layers. The last fire module is removed and the ReLUs is swapped to exponential ReLUs which decays exponentially for negative arguments. The largest change is at the output layer. The average max pooling and softmax are replaced by a so called heatmap layer. The heatmap layer takes the input from the last hidden layer and convert it to a heatmap. The heatmap displays probabilities of where pedestrians are located in the image. Figure 3.6 shows an illustration of how the heatmap is generated. For each subregion a weighted voting along the depth dimension is performed. If the voting reaches a certain threshold value the subregion is classified as a part of a pedestrian, otherwise it is not.

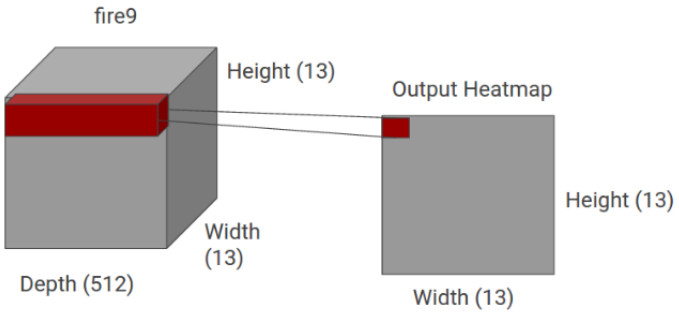


Figure 3.6: The heatmap layer in the SqueezeMap network architecture [28].

Chapter 4

Convolutional Neural Networks for Pedestrian Detection

There has been extensive research done in the field of using ConvNets for pedestrian detection [29],[30],[31],[32],[33],[34],[35],[7]. Many methods share a similar pipeline called Region-based Convolutional Neural Networkss (R-CNNs) that simplified can be described in three steps: region proposal, feature extraction and feature classification. First, regions that possibly contain a pedestrian are localised and then the features in the proposed regions are extracted and classified. In the case of pedestrian detection, the classification is binary. There is either a *pedestrian* or *not pedestrian* in each proposed region. When it comes to ConvNets for pedestrian detection the network is usually used for the feature extraction and classification but in some cases it is also used for the region proposals as well. This chapter starts with a summary of challenges with pedestrian detection (section 4.1), continues with a description of the R-CNN detection pipeline (section 4.2) and finishes with a review of related work (section 4.3).

4.1 Challenges with pedestrian detection

Pedestrian detection is one form of more generic object detection. There are many challenges that need to be taken into account when writing a pedestrian detection algorithm for autonomous vehicles. The algorithm must be generic enough to detect all pedestrians in the image but discriminating enough to only detect pedestrians. Furthermore it must be possible to implement the algorithm in an embedded system to operate in real time. Aspects as limited compute power, storage and memory bandwidth must hence be taken into consideration. Challenges with the detection are mainly due to the many appearances of pedestrians, the complex backgrounds in which they appear, different illumination and occlusion. A summary of challenges with pedestrian detection and their reasons can be found in Table 4.1.

Table 4.1: Challenges with pedestrian detection

Challenge	**Reason**
Appearance	Different articulated poses and different clothing
Background	Complex backgrounds in urban environments
Illumination	Different times on the day, different weather
Occlusion	Other objects in the environment and crowded places
Implementation	Limited compute power and memory resources

4.2 The R-CNN based detection pipeline

The R-CNN detection pipeline consists of three steps: (i) region proposal, (ii) feature extraction and (iii) feature classification. The pipeline is illustrated in Figure 4.1. A region proposal algorithm is applied on the input image and a ConvNet extracts and classifies the features in the proposed regions. The input to the pipeline is a whole image and the output is a number of bounding boxes, each representing a detected pedestrian. This section will describe each step in this pipeline in more detail.

4.2.1 Region proposals

The purpose with region proposals is to, from an whole image, extract regions that possibly contain pedestrians. Most importantly the proposed regions must contain *all* pedestrians in the given frame. If an existing pedestrian is outside the proposed regions it is impossible for the feature extractor and classifier to detect that pedestrian and it will affect the accuracy negatively and impose a safety risk. Still it is beneficial if as few regions as possible are proposed, since all the proposed regions are passed to the feature extractor and classifier which are often very computationally heavy.

The simplest and most straight forward region proposals are the Sliding Window algorithms where a window of fixed size slides over the image and for each window, features are extracted and classified. If the window classification scores above a certain threshold, it is accepted as a proposal. Since pedestrians appear in a wide

range of scales, several windows of different sizes need to slide over the image which makes these type of region proposals very inefficient.

Selective Search [36] does the region proposal in three steps using semantic segmentation. First the input image is segmented into initial subregions where each subregion belongs to at most one object. Since one object can be divided into several subregions in this step, similar regions must be combined into larger regions. This is done recursively before region proposals are extracted from these regions in the final step [36].

Aggregated Channel Features (ACF) [37] is built upon Viola and Jones [38] work for face recognition. Unlike Selective Search, ACF is based upon machine learning and is trained to search for one certain type of object instead of just searching for generic objects. The input image is described by channels. RGB and Gray-scale are for example typical colour channels but can generally be defined as a registered map of the original image and different channels can be computed with transformations. The channels are computed and vectorized into a pixel look-up table. During training a classifier is trained on the example data using AdaBoost.

In order to increase detection speed both fast R-CNN [39] and faster R-CNN [40] have been introduced. In fast R-CNN the features extracted from the region proposal are fed to the feature extraction network with the motivation that it is unnecessary to extract the same features twice. Faster R-CNN takes it one step further by applying a Region Proposal Network (RPN) to propose possible regions for a ConvNet that does the feature extractions. Features detected by the RPN are shared with the ConvNet in order to save computations. As the names suggest, fast R-CNN is faster than the ordinary approach and faster R-CNN is even faster.

4.2.2 Feature extraction and classification

For feature extraction and classification, a ConvNet is applied. Chapter 3 gives a deeper explanation of the theory behind ConvNets. The input to the feature extractor is images with proposed regions in which features are extracted and classified. The output from the classifier is one out of two labels *pedestrian* or *not pedestrian*.

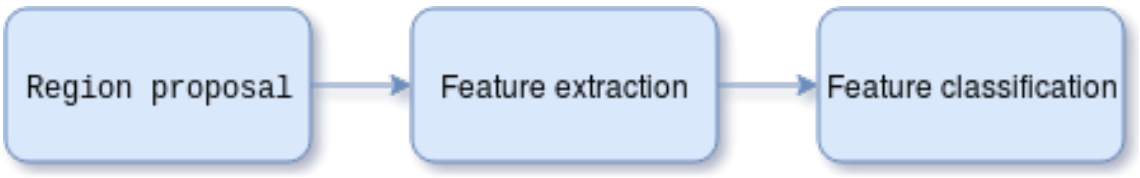


Figure 4.1: The pedestrian detection pipeline consists of three steps: (i) region proposal, (ii) feature extraction and (iii) feature classification.

4.3 Related work

Pedestrian detection algorithms can generally be divided into two categories: detection based on hand crafted features and detection based on deep convolutional features. This project is limited to only cover methods for pedestrian detection including ConvNets, but first it is worth mentioning three early works based on hand crafted features that have been influential in the field. This section first presents these three works and continues with works based on ConvNets.

4.3.1 Detection based on hand-crafted features

The first pedestrian detector was introduced in 2003 by Viola and Jones [38]. They observed that the contour of a pedestrian is characterized by abrupt changes of pixel intensity and applied haar-like features and then AdaBoost for the classification. Haar-like features categorises the image by pixel intensity and AdaBoost iteratively improves the classification by accounting for the incorrectly classification during the training. In 2005 Dalal *et al.* [41] introduced Histograms of Oriented Gradients (HOG) for feature extraction which later became a basis for more complex algorithms. HOG is a technique for object detection that counts the occurrences of gradient orientations in the image. In 2010 Felzenszwalb *et al.* [42] introduced a detector whose main idea was to describe a pedestrian as a composition of deformable parts, such as head and torso, and use HOG for the features in combination with a trained classifier.

4.3.2 Detection based on convolutional neural networks

The first pedestrian detector using ConvNets was introduced in the beginning 2013 by Sermanet *et al.* [29]. The system was pre-trained in an unsupervised manner to learn features at all levels before it was trained in a supervised manner to learn the classifier and fine-tune the features. In their end-to-end system both lower and higher levels' output were fed to the classifier, with the motivation that the classifier then receives both global shapes and structures from the higher levels and local details from the lower layers [29]. The system was only trained on a relatively small dataset, but still achieved strong results on larger datasets compared to other methods at the time. In the end of the same year Ouyang *et al.* [30] introduced a ConvNet called JointDeep that was reported to achieve the best performance on publicly available pedestrian datasets. They claimed that there are four main steps in pedestrian detection, given an input that either contains a pedestrian or not, i.e. the pedestrian does not need to be localised. The steps are feature extraction, part deformation handling, occlusion handling and classification. They coupled the deformable part model [42] with a stack of ConvNets that jointly learned all four main steps for pedestrian detection, hence the name JointDeep.

Many works have used the R-CNN based pipeline consisting of region proposal, feature extraction and feature classification. Hosang *et al.* [31] used the pipeline and in their work different region proposals were analysed. They concluded that as long as the proposals are not random the obtained quality is rather stable and

that the proposal part of the pipeline is the most time consuming. For the feature extraction and classification the small ConvNet CifarNet and the large ConvNet AlexNet were analysed. Both the small and large networks were proven to be efficient and when pre-training them on the ImageNet dataset [43] they achieved even better results. The work by Tomè *et al.* [7] is built upon the work by Hosang *et al.* and it optimised all steps in the detection pipeline. For the region proposal sliding window, Selective Search, Locally Decorrelated Channel Features (LDCF) [44] and ACF were analysed, where LDCF is an extended version of ACF. For the feature extraction and classification they analysed the ConvNets AlexNet [3] and GoogleNet [6], pretrained on ImageNet data and finetuned for pedestrian detection. They came up with a pipeline named DeepPed, consisting of LDCF for region proposal together with their finetuned version of AlexNet. They found that both AlexNet and GoogleNet performed similarly in terms of accuracy but decided AlexNet to be the most feasible due to the higher complexity of GoogLeNet. They also concluded that the region proposal was the must time consuming part of the pipeline and that ACF could be used in order to increase the detection speed. This was done by Dong *et al.* [45] with a satisfying result.

Despite the improvements of the region proposals, faster R-CNN are state of the art today [46],[47],[48]. Faster R-CNN handles the problem with slow region proposals since the RPN operates fast and features are shared with the ConvNet used for feature extraction. The work by Zhang *et al.* [46] is interesting since the features extracted by the regions are reused in the feature extraction. In the work faster R-CNN was combined with k-means clustering. K-means clustering is a method where k number of groups are found in the input data, depending on the data point similarities. In the work k-mean clustering of feature vectors from the ConvNet was used to generate initial regions and a RPN to produce accurate bounding boxes from the regions. In order to decrease the number of computations, features from the initial regions were passed as input to the RPN.

Other approaches have also been presented with purpose to increase the detection speed. Angelova *et al.* [34] approached the problem with slow region proposals by applying a fast classifier directly at the raw input images, cascaded with a tiny network and a modified version of the large AlexNet. The purpose with the tiny network was to discard as many region proposals not containing a pedestrian as possible. This in order to save computational time since AlexNet is computationally heavy and hence time consuming. Verbickas *et al.* [28] focused on decreasing the network model size and enable the model to run in real time, yet still obtaining a detection performance comparable to other, state of the art models. The model was called SqueezeMap and will be of importance in this project. Therefore, the network architecture is described in more detail in section 3.3.3.

As the field of pedestrian detection has grown, more works handling specific aspects of the problem have been presented. Several works have focused on occlusion handling [49],[50],[51]. The studies [49] and [50] addressed the problem by the use of counting. Counting is an approach where the detection algorithms is trained on data without annotation of bounding boxes but with annotations of number of pedestrians located in the image. [51] instead introduced Discriminative Parts where a pedestrian is divided into certain discriminative parts and the network is trained to detect each part. Other works have focused on the problem with large

scale variations [48],[35]. [48] used faster R-CNN and [35] addressed the problem by dividing the feature extraction between two sub-networks where one was for small scale pedestrians and the other for large scale. Depending on the size of the proposed region the results from the two networks were weighted differently and combined to receive the final classification. When it comes to detection in different light conditions, often referred to as all-day detection, many works [52],[53],[54] take advantage of extra sensors such as IR-cameras and use networks trained on both RGB and IR images. [55] on the other hand only uses RGB images and still obtain comparable results.

Conclusions From this review it can be concluded that an extensive work has already been done in the field of pedestrian detection with ConvNet. Earlier works have focused on different aspects such as improving over all accuracy or speed, decreasing model size or more specific aspects such as occlusion handling, scale variation and different light conditions. Previous implementations have been on Central Processing Units (CPUs) or Graphical Processing Units (GPUs) and there is a lack of systems running in real-time in embedded systems.

Chapter 5

Convolutional Neural Networks and Field Programmable Gate Arrays

---

As stated in previous chapters, ConvNets achieve strong results for pedestrian detection, which is important in applications such as autonomous vehicles. However, autonomous vehicles require the systems to be embedded. Embedded systems are prone to have limited resources such as power, memory bandwidth and computational power. ConvNets are known for being computationally heavy and demand large memory space and memory bandwidth, which implies some struggle in the implementation. Usually high performing ConvNets are implemented on GPUs but lately FPGAs have started to compete with the GPUs. FPGAs are promising but only have small onchip memory, which puts demands on smaller and more energy efficient ConvNets and smarter acceleration strategies. Several works [56],[57],[58] have despite these challenges accelerated ConvNets on FPGAs successfully by taking advantage of the inherent parallelism of ConvNets and the reconfigurable characteristics of FPGAs.

When accelerating ConvNets on FPGAs there are mainly two aspects that can be focused on. The first is the software aspect, where the goal is to compress the network in order to minimise memory requirements and the required number of operations. The second is the hardware aspect, where the goal is to reuse as much data as possible and accelerate the convolution operations through parallelisation. Since this project is about ConvNets, the focus will be on the software aspect of the acceleration. This chapter first introduces the reader to FPGAs and how they can be used for ConvNet-acceleration compared to GPUs (section 5.1) and then describe how ConvNets can be optimised for FPGA-acceleration (section 5.2).

5.1 Field Programmable Gate Arrays

An FPGA is a semiconductor device whose physical functionality can be changed in the field, i.e. after it has been manufactured. The code written to an FPGA makes real physical connections with wires to perform the desired functionality [59]. This enables the user to not only design the software, as the case is with micro controllers, but also design the hardware. Since code always is executed faster on hardware than software, FPGAs have the potential to perform operations several times faster than hard-wired processors [60]. Simply described, an FPGA consists of millions of logical cells that physically form arrays. FPGAs of today however, have been complemented with something called hard Intellectual Property (IP) for the most commonly used functions such as DRAM controllers, clock generators and multiply-accumulators. These pre-defined blocks of hardware are not reconfigurable as the rest of the FPGA, but since they are commonly used, they simplifies the design process [59].

Table 5.1 shows a summary of advantages and disadvantages with FPGAs compared with GPUs for the use of ConvNet-acceleration. First of all, FPGAs can exploit the inherent parallelism of ConvNets. GPU implementation can be parallelised in such a way that a large number of images can be processed in parallel. This is an advantage during training where all data is given from the beginning. In real time applications on the other hand, the images must be processed one by one, in order to minimise latency. This implies an irregular parallelism and hence the parallelisation offered by GPUs is not optimal. FPGAs are more flexible and offers a pipelined parallelism which can operate different, dependent steps of computations concurrently on different threads [56]. Furthermore the hardware execution of code offered by FPGAs are faster than software execution since software execution requires instructions to be fetched and cued up, operations to be done and results to be sent to memory. FPGAs have also been shown to be more energy efficient than GPUs in terms of

Table 5.1: Advantages and disadvantages with FPGAs for ConvNet-acceleration.

Advantages	**Explanation**
Pipeline parallelism	Output from one step is streamed as input to the next step, while execution of the steps is overlapping.
Speed	Enable faster execution since hardware execution is faster than software execution.
Energy	Are more energy efficient in terms of operations per watt compared to GPUs.
Disadvantages	**Explanation**
Floating point operations	Do not have built in IPs for floating point operations.
Memory	Limited on-chip memory and limited bandwidth to off-chip memory.
Hard to program	Require knowledge about the hardware and HDL.

operations per watt.

On the other hand, ConvNet-computations often consist of a large number of floating point operations since they often are designed for GPUs which, unlike FPGAs, are native floating-point processors. Floating points also requires more memory than fixed points and both on chip memory and off chip memory are limited on FPGAs. Earlier a deep understanding of the hardware and knowledge in an Hardware Description language (HDL) was required in order to program an FPGA. Lately, this has started to change when a number of frameworks for translating high level programming languages to HDL have been introduced.

5.2 Optimisation of ConvNets

Since the introduction of AlexNet it has been a trend in making the networks deeper with more parameters. The networks have become more accurate but also much larger and more computationally heavy. Lately it has been a trend in making the networks more efficient while maintaining the same accuracy [57]. Two commonly used methods for improving the efficiency are to use more compact data types and take advantage of sparse matrix multiplication, by introducing more zeroes in the network.

Compact data types Traditionally, network weights are represented by 32-bit floats, which is not a problem to handle for a GPU since the floating point IP is a fixed part of the chip architecture. In an FPGA the logic elements are not fixed, so it is advantageous to use fixed point multiplication instead. Then the number of multiplications that can be implemented increases. Furthermore, using more compact data types to represent the weights both reduces memory requirements and power consumption [61]. Researchers have shown that it is possible to use 8, 4 or 2 bits without decreasing accuracy distinctly [62],[63],[64]. Binary networks using only 1 bit for representing the weights have also been investigated [65], which decreases the memory requirement drastically, but are not yet as accurate as 32-bit floats.

In general there are two ways to receive a quantised ConvNet. Either the network is initialised and trained with fixed point data from the beginning or it trained with floating point data, which is then converted to fixed point data. It can be argued that the former approach implies higher accuracy but it is not always suitable to use that approach. In many cases it is not possible to train the network from scratch due to a limited amount of data. In such cases a pre-trained model must be used and most available pre-trained models are unfortunately with floating point data.

Sparse matrix multiplication The majority of the operations done in a ConvNet are matrix multiplications. Since sparse matrix multiplication is less computationally heavy than ordinary matrix multiplication, it is advantageous if larger portions of the matrices consist of zeros. The number of zeroes can be increased by using ReLU as activation function, since it puts all negative values to zero, or by

using "pruning". Pruning means that all weights below a certain threshold value are set to zero [63]. Pruning works since small weights have little effect on the total network predictions and can therefore be neglected.

Chapter 6

Datasets

There exist a number of pubic datasets containing images of persons and pedestrians in different poses, environments and with different amount of occlusion. The INRIA person dataset [41] is a comparatively small dataset containing persons in upright positions. The Caltech pedestrian dataset [66] is one of the largest and it is also one of the most commonly used for evaluation of pedestrian detection algorithms. Evaluating an algorithm on the Caltech pedestrian dataset hence enables a fairly straightforward comparison with other, state of the art algorithms.

When it comes to the task of more generic object classification and detection, there exist even more datasets. The most extensive and most commonly used for comparison of object classification and detection algorithms is the ImageNet dataset [43].

In this project the INRIA person, Caltech pedestrian and ImageNet datasets are of importance and therefore this chapter presents them in more detail.

6.1 INRIA person dataset

The INRIA person dataset was collected as a part of a research work conducted by Dalal [41] who thought the datasets existing at the time were not challenging enough. The images in the dataset mainly comes from three sources: from the GRAZ 01 dataset, from a personal camera and from the web using Google images. The dataset both contains full images that are annotated with person location and images that are cropped around the persons. The annotations only indicates persons, there are no labels for ambiguous cases or any information about occlusion.

6.2 Caltech pedestrian dataset

The Caltech pedestrian dataset was collected by filming 10 hours of 30Hz video (10^6 frames) from a vehicle driving through regular traffic in urban environment in 11 different sessions. Out of the 10 hours long video, approximately 137 minutes long segments (250 000 frames) where annotated by hand.

In the dataset, pedestrians are marked with rectangular bounding boxes covering their whole appearance. Occluded pedestrians are marked with two bounding boxes: one for the visible part and one for the full extent. Crowded places where it is hard to distinguish between different pedestrians are marked as 'people' and areas where it is unclear if there are pedestrians or not are marked as 'pedestrian?'. In average each pedestrian is visible for 5 seconds and is partially occluded for 3.5 seconds. In total the dataset contains 350 000 bounding boxes with 2300 unique pedestrians. 50% of the frames contain no pedestrians, 30% contain two or more and in average each frame contains 1.4 pedestrians. A summation of the dataset is shown in Table 6.1 [66].

In their paper, Dollar *et al.* [66] propose four different ways in which the Caltech pedestrian dataset can be used for training and evaluation. This in order to allow different algorithms to be compared easily. The dataset is divided into 11 different sessions (S0-S10), filmed in different neighbourhoods. Using parts of this sessions or external data for training and parts of the sessions for evaluation, results in four evaluation scenarios (the same data can never be used for both training and evaluation since that would imply a positive bias to the result):

1. Train on any external data, test on S0-S5.
2. Train on any external data, test on S6-10
3. Perform 6-fold cross validation using S0-S5. In each phase use 5 sessions for training and the 6^{th} for testing, then merge and report results over S0-S5.
4. Train on S0-S5, test on S6-S10.

The standard way of training is to use every 30^{th} video frame which results in a total of 4 250 frames. Instead of this standard set up, Dollar *et al.* proposes in [37] that every 3^{rd} frame can be used for training to increase the number of training data. This set up is referred to as Caltech 10x.

Table 6.1: Summary of the Caltech pedestrian dataset statistics [66].

total # frames	1000k
# labeled frames	250k
# frames w pedestrians	132k
# bounding boxes (BB)	350k
# occluded BB	126k
# unique pedestrians	2300
average pedestrian duration	5s
average # labels per frame	1.4

6.3 ImageNet dataset

ImageNet is an ongoing research project with the goal to provide an easily accessible image database [43]. The database contains more than 14 million annotated images whereof approximately 1 million have bounding box annotations. The images are divided into 27 high level categories which in turn are divided into 21 000 subcategories. The database was constructed by collecting images on the web and annotate them by hand.

The ImageNet database is frequently used for training and validation of object classification and detection algorithms due to the large number of annotated data and the wide variety of the object classes. A subset of the database is used in ILSVRC in which state of the art methods for image classification, object detection etcetera are compared with each other.

Chapter 7

Design and implementation

This chapter describes how the design and implementation is conducted. It starts with describing the network design (section 7.1) and continues with what hardware and software that is used (section 7.2). Then the steps in the implementation are described (section 7.3) and finally the data preprocessing (section 7.4) and evaluation metrics (section 7.5) are presented.

7.1 Network design

The network design is done with the conclusions from the literature study in mind. The final goal is to have a network that can be implemented on an FPGA in an autonomous vehicle and therefore there are a number of challenges that need to be taken into account. Table 7.1 displays the challenges and the root to them, that is either the functionality or the implementation. Challenges linked to the functionality are mainly the detection speed and the detection accuracy. Since it was concluded that the region proposal is the most time consuming part in the R-CNN detection pipeline this design aims to skip the region proposal and instead process whole frames in a similar way as the SqueezeMap network (described in section 3.3.3). This is done by the realisation of a heatmap layer which outputs a prediction for each part of the image whether a pedestrian is there or not. To achieve an as high accuracy as possible, an already existing network architecture are used together with transfer learning. Then the network has already learnt low level features and is proved to achieve high accuracy on at least generic image classification.

The other root to challenges is the implementation itself. Here the challenges mainly lay in the limited memory resources, limited power supply and the limited support for floating point operations in FPGAs. For this reason, it is desirable to have a small network where as few operations as possible are performed. Due to this the network architecture SqueezeNet is chosen (described in section 3.3.2). It achieves high accuracy on the ImageNet dataset and as the name suggests the network is squeezed down to have a low model size. SqueezeNet is released in two versions, version 1.0 and 1.1. Both versions achieve the same accuracy on ImageNet classification but due to some minor changes in the first convolutional layer and the pooling layers, SqueezeNet version 1.1 needs 2.4 times less computations than version 1.0.

In order to meet the requirements on small model size and a minimised number of floating point operations two promising strategies are to use pruning and quantisation (see section 5.2). The authors of Deep Compress [63] have compressed SqueezeNet and their work in combination with a heatmap layer from this work is one solution to be able to implement a ConvNet for pedestrian detection on an FPGA.

Table 7.1: Challenges in the network design and their proposed solution.

Root	Challenge	Proposed solution
Function	Speed	Skip region proposal
	Accuracy	Use existing network and transfer learning
Implementation	Memory	Decrease model size
	Power	Decrease number of computations
	Floating points	Decrease number of floating point operations

7.2 Hardware and software

Implementing deep learning models from scratch can be a tremendous work. Therefore, frameworks with the most commonly used routines implemented are often being

used, both in academic and industrial work. One commonly used framework is called Caffe [67] which is used in this project as well. Deep learning models requires a lot of computational power which puts high demands on the hardware being used. An alternative to buying all required hardware is to perform the computations in the cloud.

This section presents Caffe (section 7.2.1) and Amazon Web Services (AWS) which is a cloud services platform (section 7.2.2).

7.2.1 Caffe

Convolution Architecture For Feature Extraction (Caffe) is an open source deep learning framework developed by Berkeley AI research [67]. It is written in C++ with Python and Matlab bindings which makes it user friendly. The workflow when working with Caffe is visualised in Figure 7.1. In the first step all data that is going to be used is preprocessed and prepared. This includes saving the data in a format that can be read by Caffe, for example as a Lightning Memory-Mapped Database (LMDB) or in Hierarchical Data Format (HDF). In the second step the model itself is being defined. That is defining all layers and all layer parameters. In the third and fourth steps the solver and training definitions are written. This includes defining learning rate, batch size (how many images are processed at the same time) and other parameters that are needed for solving and training.

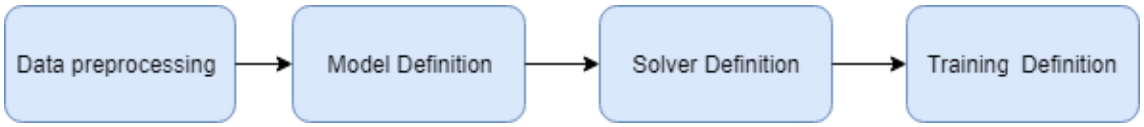


Figure 7.1: Working with Caffe consists of four steps. Data preprocessing, model definition, solver definition and training definition.

There are several reasons why Caffe is chosen to be used in this project. First of all, the most commonly used layers such as convolutional layers and pooling layers are already implemented. Caffe supports NVIDIA CUDA Deep Neural Network library (cuDNN) which is a GPU-accelerated library of primitives for deep neural networks [68]. NVIDIA cuDNN provides optimised implementations of the layers through CUDA which is a parallel computing platform that gives direct access to the GPU's parallel computational elements. This speeds up the training of the network since a lot of computations can be performed in a parallelised manner. There also exist frameworks for implementing trained Caffe models in embedded systems. The SDSoC environment developed by Xilinx [69] for example supports scheduling of pre optimised neural network layers on FPGAs.

Caffe is open source which makes it possible to do changes according to specific needs. It is possible to write custom layers in both Python and C++. Python layers are often well suited for testing since there is no need to recompile the whole Caffe installation as is the case with C++ layers. However, Python layers do not support multi GPU execution which makes it desirable to write the final layer implementation in C++.

In the academic research Caffe is commonly used for the implementation of ConvNets. Many authors, including the authors of networks like AlexNet and SqueezeNet, make their network models and weights from training on ImageNet data publicly available in the so called Caffe model zoo. This enables others to validate, test and use the models in their research. In this project SqueezeNet is of interest and Iandola *et al.* [27] have made the SqueezeNet model with weights from training on ImageNet publicly available in the Caffe model zoo. Songhan *et al.* [63], the authors of Deep Compress, have also made a deep-compressed version of SqueezeNet publicly available in the Caffe model zoo which can be used for future work.

7.2.2 Amazon Web Services

Available hardware is not always enough for certain computations and it is not always possible to buy hardware that is good enough. In such cases cloud computing is an option. Then the computations are performed in the cloud instead of on the local hardware. The cloud offers compute power, database storage or what else could be needed. Physically the resources can be located anywhere in the world and are often shared among a large number of users. AWS [70] is a cloud services platform where the costumer pays for the time the services are used and no more. The hardware provided by the cloud is reached from the local disc via a Secure Shell (SSH) [71]. SSH is a network protocol that is used to securely connect to a network server via an insecure network.

One part of AWS is Amazon Elastic Compute Cloud (EC2) which provides computing capability in the Amazon cloud. In Amazon EC2 one or several virtual computing environments can be launched, called instances. Either an instance can be launched "as is" without operating system and other software or with a preconfigured template. These templates are called Amazon Machine Images and come with operating system and other software pre installed. One of these AMIs is called Deep learning for Ubuntu and comes with Ubuntu along with Caffe and other frameworks for deep learning. The Caffe installation is CUDA and CuDNN compatible which enables faster execution on the GPUs.

7.3 Implementation

The implementation is conducted according to the following steps. Each step is described in more detail below.

1. Classification on the INRIA person dataset.
2. Implementation and testing of a heatmap layer
3. Detection on the Caltech pedestrian dataset

Initially the network is trained and tested for pedestrian classification on the normalised images in the INRIA person dataset. The purpose is to see how well the network performs on the task at hand and to receive weights that can be used for transfer learning in later steps of the implementation. Both SqueezeNet version 1.0

and version 1.1 are used and compared with each other. All layers except from the last are kept the same. The last layer is changed from classifying 1000 labels to only classify two labels instead. The training runs both locally on the CPU of a laptop and in the cloud on AWS in order to compare running times and conclude whether it is possible to do the training locally on the CPU or not.

In the second step a heatmap layer is implemented. Since there is no implemented heatmap layer in Caffe a custom python layer is written. In the forward pass all elements $a_{i,j}^{l+1}$ in the output feature map are given by

$$a^{l+1} = \sum_{n=0}^{N} a_{n,i,j}^{l} \cdot w_{n,i,j} + b_{n,i,j} \tag{7.1}$$

where N is the number of channels of the input feature map, $a_{n,i,j}^{l}$ are the elements in the input feature map, $w_{n,i,j}$ are the layer weights and $b_{n,i,j}$ are the biases. A visual representation of the layer is shown in Figure 7.2. During the backward pass the gradient of the loss function with respect to each of the weights needs to be known. The gradient is calculated by applying the chain rule: all derivatives that "leads to" a certain weight are multiplied with each other. Since the heatmap layer is the final layer in the network, the derivative of the cost function E with respect to the weights $w_{n,i,j}$ in the layer is simply given by

$$\frac{\partial E}{\partial w_{n,i,j}} = a_{n,i,j} \tag{7.2}$$

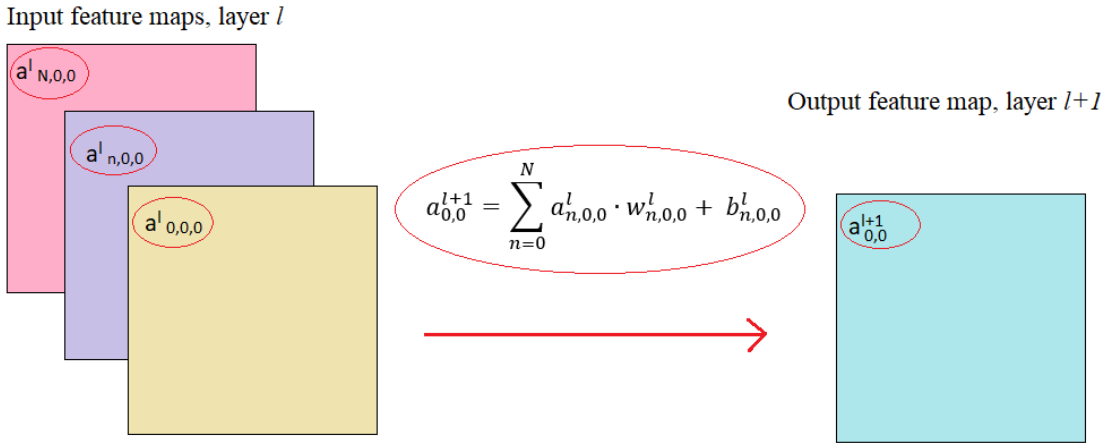
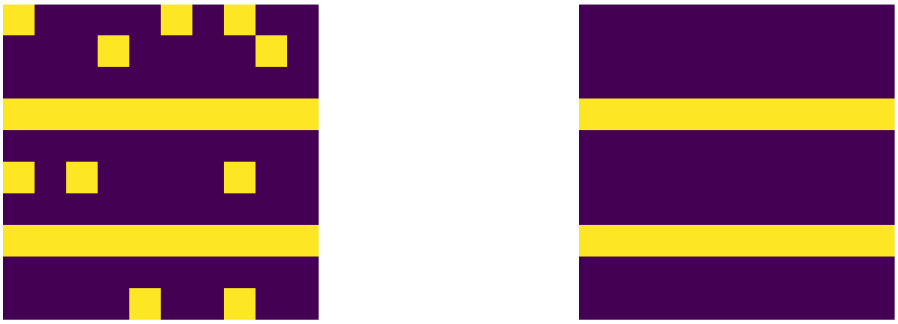


Figure 7.2: Illustration of a heatmap layer.

Before merging the heatmap layer with the pedestrian classification network from step 1, the heatmap layer is tested on a simple problem together with a small network. The network only consists of one so called fire module, described in section 3.3.2. The fire module first compresses the input image size and then expands it again, giving an output with the same spatial dimensions as the input. The input image is of dimension 10x10 where the goal is to extract vertical lines. When working with the heatmap layer the classification is no longer binary. Instead, the label for

an image is a matrix, where each position in the matrix describes what is in that position in the image. Figure 7.3a shows an example of an image and 7.3b shows the label matrix, indicating where the two vertical lines in the image are located.



(a) (b)

Figure 7.3: Images for testing the heatmap layer (a) displays the input image and (b) displays the label matrix.

Finally the the two first steps is merged together to get a pedestrian detection network. The network is trained on the Caltech pedestrian dataset which consists of images of size 480x640. When the images are fed through the network their spatial dimensions after each layer change according to

$$w' = \frac{w - f + 2p}{s + 1} \tag{7.3}$$

where w' is the output size, w the input size, f the filter size, p the padding and s the stride. Feeding an image through all layers except from the the last global pooling layer in the SqueezeNet version 1.1 network (illustrated in Figure 3.5) hence gives a feature map with dimension 29x39. The label matrix must therefore also be a matrix with with dimension 29x39.

7.4 Data preprocessing

In the first and third step of the implementation the INRIA person dataset and the Caltech pedestrian dataset are used. These datasets need to be preprocessed which are described in this section. In the second step a dataset with images similar to the one in Figure 7.3a are constructed and used. These images do not need any preprocessing other than to be saved in LMDB format.

7.4.1 INRIA person dataset

In the first step of the implementation, a ConvNet classifies whether an image is a pedestrian or not. For this purpose cropped images from the INRIA person dataset,

normalised to a size of 96x160 pixels are used. Since the dataset in its original format contains a comparatively small amount of data all images are mirrored around the vertical centreline, as illustrated in Figure 7.4. The mean value of each channel (RGB) is determined and subtracted from the input image to centre data around zero and remove illumination differences which are not of interest. The positive images are labeled with 1 and the negative with 0 and the images together with the labels are shuffled and saved in LMDB format.

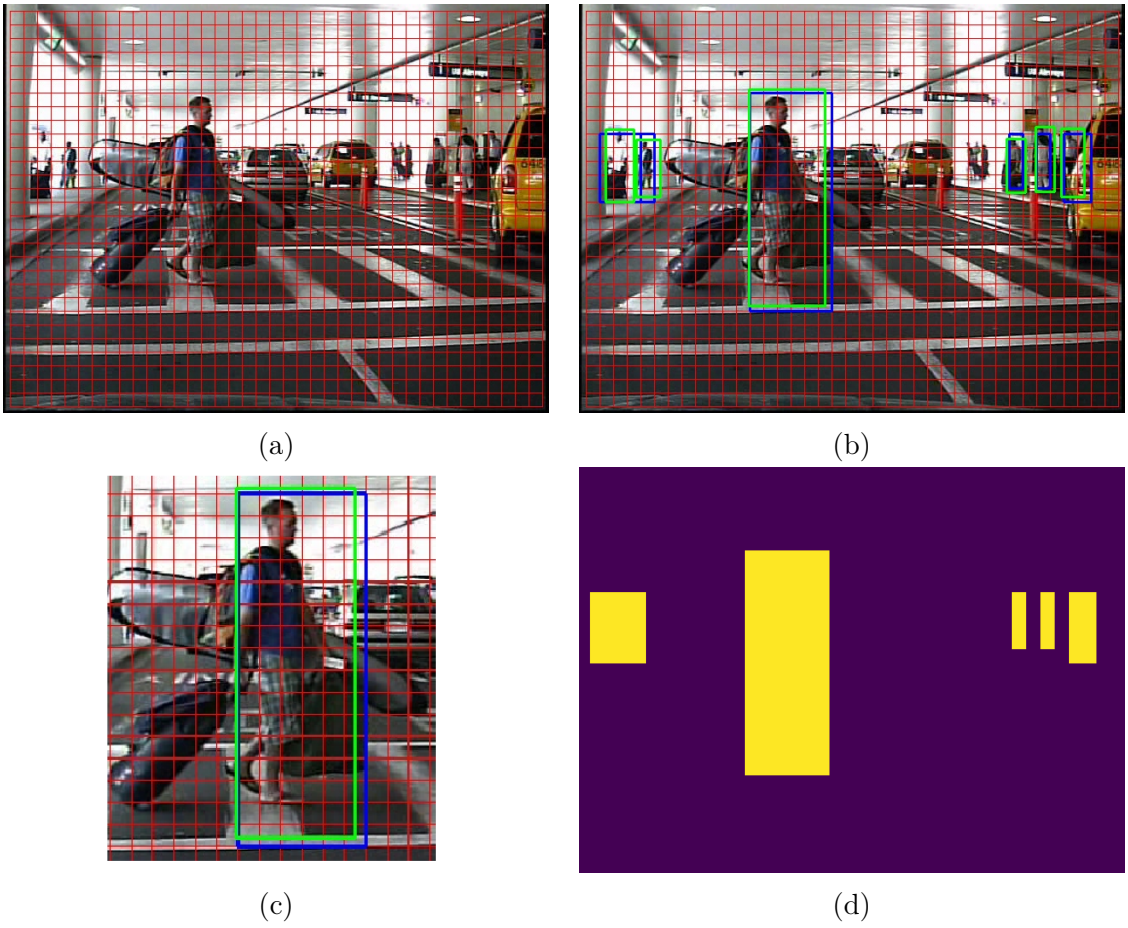
Figure 7.4: Mirrored image from the INRIA person dataset.

7.4.2 Caltech pedestrian dataset

As with the INRIA person dataset the images in the Caltech pedestrian dataset are also subtracted by the mean value of each channel. Labeling the images in the Caltech pedestrian dataset is not as straight forward as with the INRIA person dataset. Since the output from the heatmap layer is a matrix of size 29x39 where each element indicates whether that position is a pedestrian or not, the label matrix should be of the same size and with the same structure. The existing labels for the dataset consist of bounding boxes around the pedestrians where the coordinates for the top right and bottom left corners are given. In order to create the label matrix, the original images are divided into a grid of size 29x39 with grid boxes of size 16 pixels. Since the image size is 640x480 pixels, 8 pixels are discarded along each side of the image. An example image and the grid can be seen in Figure 7.5a. Since the annotated bounding boxes do not necessarily intersect exactly with the grid, they need to be adjusted to fit the grid. This is done by changing the position of each side of the bounding box to the closest grid line. This implies that the position of each side of the adjusted bounding box is at maximum 8 pixels from the original annotation. Figure 7.5b and 7.5c shows the original annotated bounding boxes of all pedestrians in the frame with a height above 50 pixels in green and the adjusted bounding boxes in blue. Figure 7.5d visualises the final ground truth label matrix where yellow indicates that there is a pedestrian at that position and blue indicates that there is no pedestrian.

Two strategies for sampling and labeling images from the original videos are used. The first is referred to as Caltech 10x, meaning that every third frame in the original videos are used for training and testing giving totally 42 782 images for training and 40 465 images for testing. Figure 7.6a displays the percentage of nonzero values in

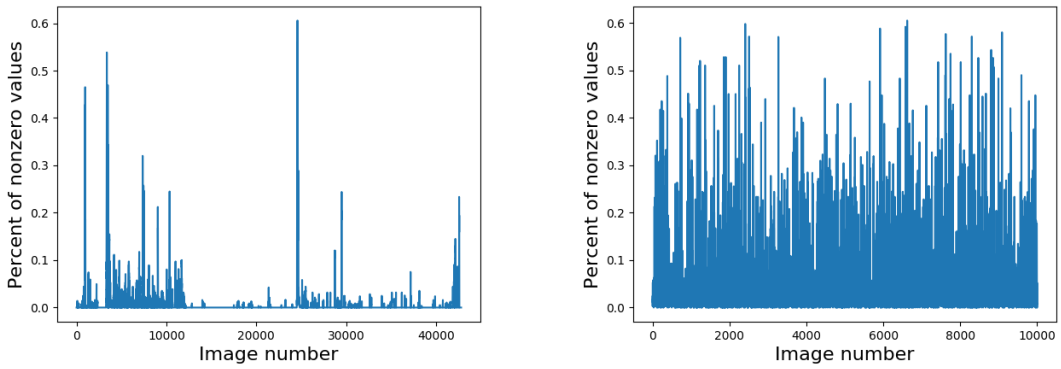
the label matrix for each image in the training set sampled according to Caltech 10x. 78% of the images does not contain any label *pedestrian*. The second strategy is to randomly select 10 000 images containing at least one pedestrian with height larger than 50 pixels. If the same image is chosen twice a mirrored version of that image is used in the set. All pedestrians with a height larger than 50 pixels that are not occluded are labeled *pedestrian*. Figure 7.6b displays the number of nonzero values in each image. Labels for smaller or occluded pedestrians, the labels *pedestrian?* and *people* are discarded in both strategies. As with the INRIA person dataset the images are saved in LMDB format.



(a) (b)

(c) (d)

Figure 7.5: An example image from the Caltech pedestrian dataset. (a) the grid is displayed on top of the image, note that 8 pixels along each side are discarded (b) bounding boxes around all pedestrians with a height above 50 pixels. The original annotated bounding boxes are marked with green and the adjusted are marked with blue (c) zoom of one pedestrian with both the original bounding box (green) and adjusted bounding box (blue) marked (d) the corresponding labelling matrix. Yellow indicates that there is a pedestrian at that position and blue indicates that there is no pedestrian.



(a) Images sampled according to Caltech 10x.

(b) Images sampled randomly as described in the text.

Figure 7.6: Percentage of nonzero grids in the label matrices for both sampling methods described in the text.

7.5 Evaluation metrics

Classification on the INRIA person dataset is binary which makes it straight forward to measure the accuracy. It is given by the ratio

$$\text{accuracy} = \frac{\#\text{ correct labels}}{\#\text{ images}} \tag{7.4}$$

This measure of accuracy is also called top-n accuracy which means the accuracy that the ground truth label is within the n most probable predictions. Here $n = 1$.

First the dataset is divided into two sets, one for training and one for testing. The network is tested continuously on the test set during training and the loss and accuracy is displayed. Then 6-fold cross validation is performed. The dataset is divided into six sets where five sets are used for training and the sixth for testing. This is done six times so that all sets are used as test sets. All training sets are also validated with a validation set that is totally new for the network and does not come from the INRIA person dataset. The validation set consists of 50 images collected from the web where 30 are positives and 20 are negatives.

The initial implementation of the heatmap layer is evaluated by measuring the Euclidean loss E, given by

$$E = \sum_{i,j} (a'_{i,j} - a_{i,j})^2 \tag{7.5}$$

where $a'_{i,j}$ is the predicted output at position i, j and $a_{i,j}$ is the ground truth at position i, j. For this implementation no accuracy is measured since the Euclidean loss gives enough information whether the heatmap layer works or not.

For detection on the Caltech pedestrian dataset the Euclidean loss is evaluated during training. Due to bad results and time limitations no further accuracy is measured.

Chapter 8

Results

This chapter presents the results obtained in the project. The layout follows the three steps described under implementation in section 7.3. The first section presents the results from classification on the INRIA person dataset (section 8.1), the second section the results from the initial heatmap layer implementation (section 8.2 and the third section the results from detection on the Caltech pedestrian dataset (section 8.3).

8.1 Classification on the INRIA person dataset

In this section results from classifying images from the INRIA person dataset are presented. Table 8.5 concludes the training parameters that are used if nothing else is stated. The network weights are either initialised by transfer learning or randomly. By transfer learning means that weights from pre training on the ImageNet dataset are used and random weight initialisation means Xavier initialisation. The training is conducted through 1000 iterations with a batch size of 512 using SGD and momentum 0.9. The initial learning rate is 0.001 and decreases with 0.0001 for each iteration.

Table 8.2 shows the average time for one forward pass, one backward pass and 1000 full passes (forward and backward passes with weight updates) in CPU VS GPU mode for SqueezeNet version 1.0 and 1.1 respectively. It can be seen that the time for training SqueezeNet version 1.0 is about the double of the time to train SqueezeNet version 1.1. It can also bee seen that it is more than 20 times faster to train both models on GPU than on CPU.

Table 8.1: Training parameters used for classification on INRIA person dataset if nothing else is stated.

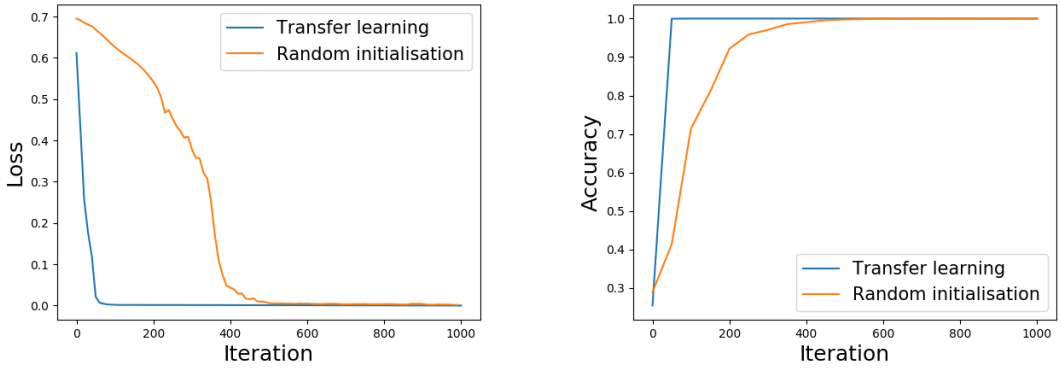
Number of iterations	1000
Batch size	512
Momentum	0.9
Initial learning rate	0.001
Weight decrease in each iteration	0.0001

Table 8.2: Average time for one forward pass, one backward pass and 1000 full passes in CPU VS GPU mode for SqueezeNet version 1.0 and 1.1.

Version	Mode	Forward pass [ms]	Backward pass [ms]	1000 full passes
1.0	CPU	3887.81	5671.52	2h 39min 20s
	GPU	128.408	293.507	7min 2s
1.1	CPU	1915.12	2707.07	1h 17min 2s
	GPU	68.4985	155.089	3min 43s

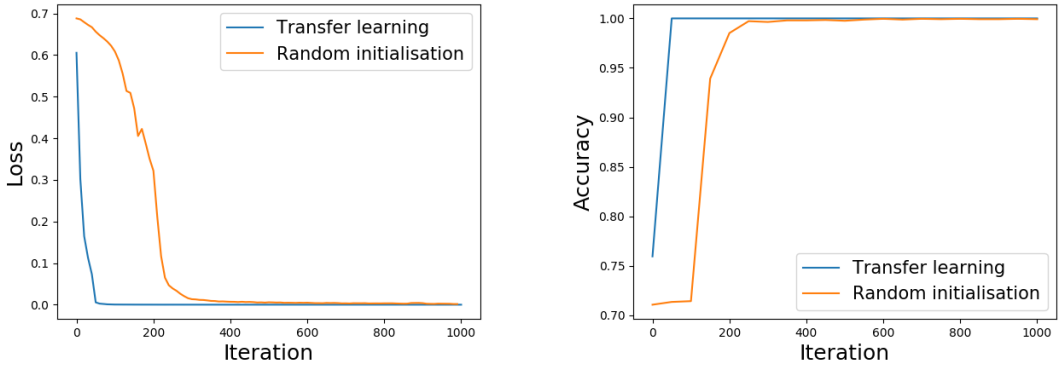
Figures 8.1 and 8.2 show the loss and accuracy when training for 1000 iterations on the INRIA person dataset for SqueezeNet version 1.0 and version 1.1 respectively. The dataset is divided into two subsets, one for training and one for testing and the loss and accuracy are displayed for the test set. The blue lines represent transfer learning and the orange lines represent random weight initialisation. It can bee seen that the loss decreases to zero and the accuracy increases to one in all cases. When using transfer learning the loss decreases and the accuracy increases faster than when using random initialisation but reaches the same accuracy after a number of iterations. Comparing the both versions it can bee seen that version 1.1 is slightly faster than version 1.0 both with and without transfer learning.

In Table 8.3 the mean accuracy and standard deviation after performing 6-fold cross validation and 1000 iterations of training is displayed. Both networks classify all images in all test sets correctly and therefore the mean accuracy is 1 with standard deviation 0. Both networks also classify all images in the validation set correctly.



(a) Training loss (b) Training accuracy

Figure 8.1: Loss and accuracy during training for SqueezeNet version 1.0 with weights initialised via transfer learning (blue lines) and randomly (orange lines).



(a) Training loss (b) Training accuracy

Figure 8.2: Loss and accuracy during training for SqueezeNet version 1.1 with weights initialised via transfer learning (blue lines) and randomly (orange lines).

Table 8.3: Mean accuracy and standard deviation for classification on the INRA person dataset

Version	Weight initialisation	Mean accuracy	Standard deviation
1.0	Transfer learning	1	0
	Randomly	1	0
1.1	Transfer learning	1	0
	Randomly	1	0

8.2 Heatmap layer implementation

Figure 8.3 shows the Euclidean loss for 1000 iteration when training the small network on the simple problem described in section 7.3. All training parameters are the same as described in Table 8.1 and the weights in the heatmap layer are initialised randomly from a normal distribution and the rest of the weights are initialised by Xavier initialisation. The loss decreases steadily and goes towards zero.

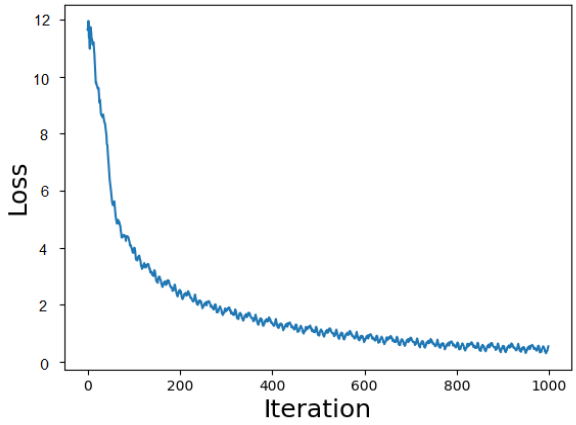


Figure 8.3: Loss during training of the heatmap layer

8.3 Detection on the Caltech pedestrian dataset

In this section results from detection on the Caltech pedestrian dataset are presented. Table 8.4 concludes the training parameters that are used if nothing else is stated. The network weights are initialised by transfer learning from training on the INRIA person dataset. A batch size of 1 is used together with SGD and momentum 0.9. The initial learning rate is 0.001 and decreases with 0.0001 for each iteration. The small batch size is chosen due to the large memory requirements when using images of size 480x640 pixels and labels of size 29x39.

Table 8.4: Training parameters used for detection on the Caltech pedestrian dataset if nothing else is stated.

Batch size	1
Momentum	0.9
Initial learning rate	0.001
Weight decrease in each iteration	0.0001

The loss during training on the Caltech 10x sampled image set is displayed in Figure 8.4. The loss decreases rapidly during the first iterations and then stays on relatively low values. Figure 8.5 is a zoomed image from Figure 7.6. It shows the percent of nonzero values in each of the first 1000 images of the training set. These are the images used during the first 1000 iterations since the batch size is 1. Comparing Figure 8.5 with the zoomed graph in Figure 8.4 it can be seen that they are rather similar.

When the percent of nonzero values in an image increases, the loss increases as well. This implies that the network always predicts *no pedestrian* in all images.

Figure 8.6 shows the network output when the input contains no pedestrians and when it contains four pedestrians of different sizes respectively. The left rectangles are the label matrices where the yellow areas correspond to the label *pedestrian* and the blue areas correspond to the label *no pedestrian*. The right rectangles are the network output. It is zero everywhere meaning that the network does not detect any pedestrians.

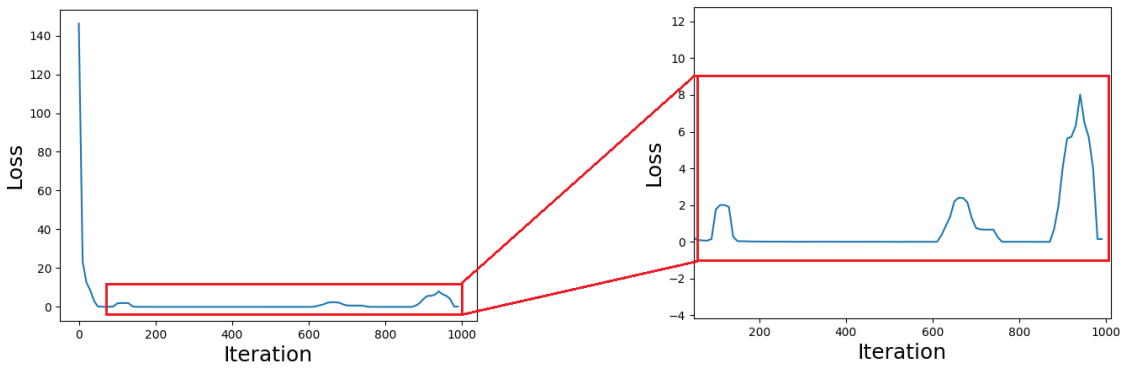


Figure 8.4: Loss during training on the Caltech pedestrian dataset when the data is not shuffled.

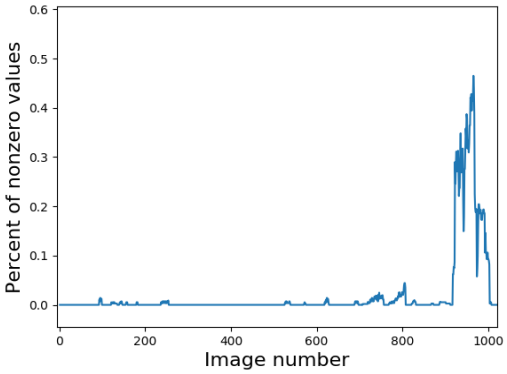


Figure 8.5: Percent of nonzero values in the first 1000 images of the training set

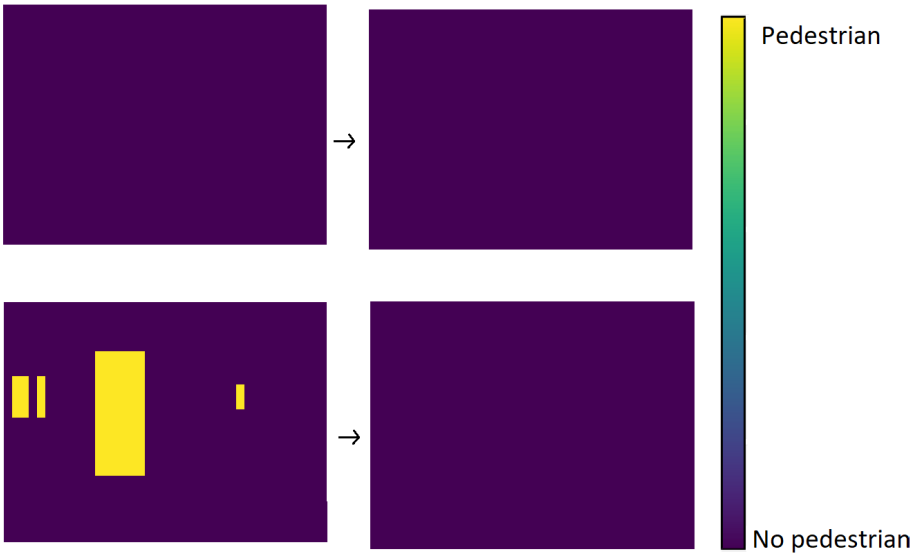


Figure 8.6: Ground truth and network prediction on image from Caltech pedestrian dataset not containing any pedestrians and image containing four pedestrians respectively.

It was decided to train the network on a very small dataset in order to see if it could learn from the images in the Caltech dataset. The small dataset consists of two randomly chosen images where one contains pedestrians and the other does not contain any pedestrians. Figure 8.7 shows the loss during this training. It can bee seen that the loss rapidly decreases during the first iterations. After 1000 iterations it has decreased from 142 to 3.53 and after 100 000 iterations it has decreased to 0.18. Figure 8.8 is a visual representation of the network output after 1000 and 10 000 iterations respectively. After 1000 iterations the output is a bit blurry compared to after 10 000 iterations where the output at first sight may look identical to the ground truth. Looking more carefully it can bee seen that there are minor colour changes in the output resulting in the loss. Feeding images that are not seen by the network before does not result in any detections since the network is heavily overfitted.

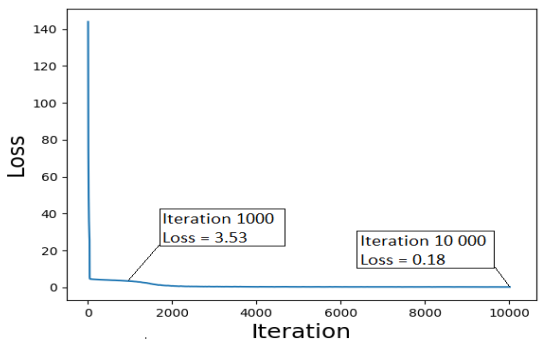


Figure 8.7: Loss during training on 2 images from the Caltech pedestrian dataset.

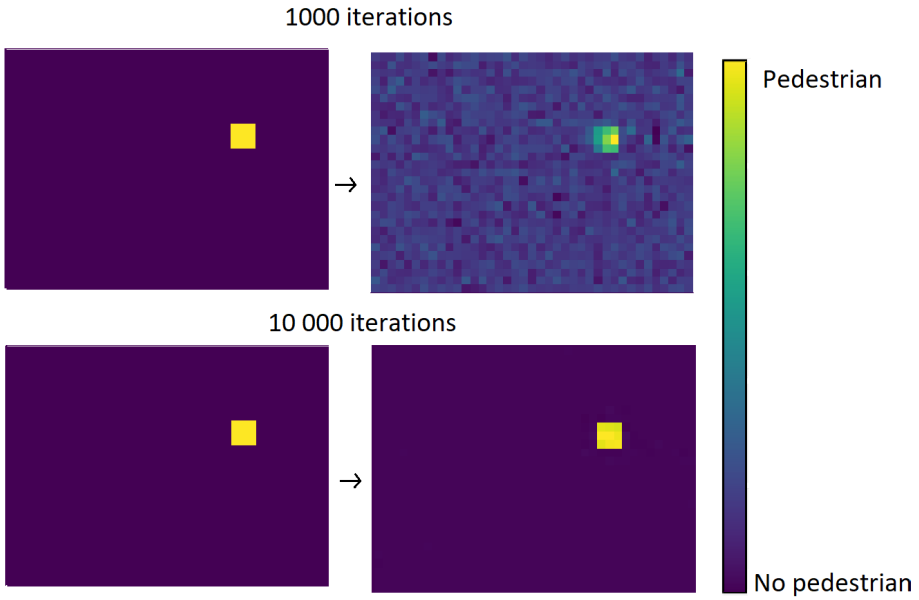
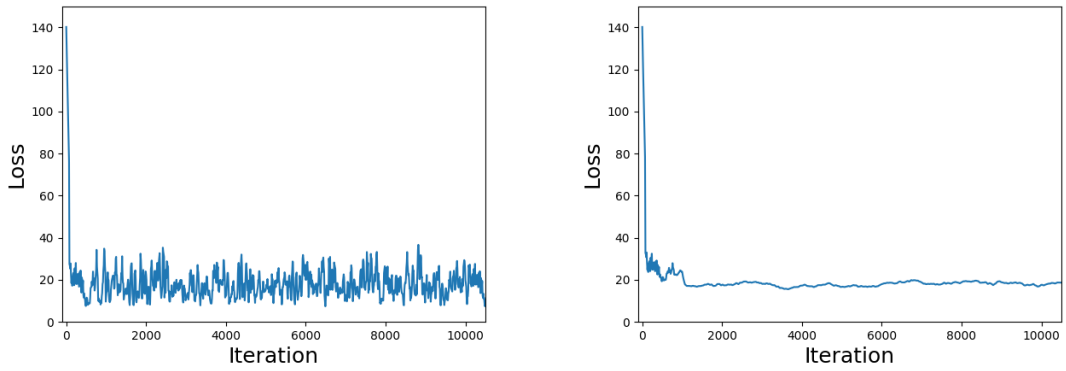


Figure 8.8: Ground truth and network prediction on image from Caltech pedestrian dataset after 1000 and 10 000 iterations respectively.

Figure 8.9 shows the loss during training on the randomly sampled image set. Only the 10 000 first out of 50 000 conducted iterations are displayed since the same pattern repeats during all iterations. Figure 8.9a shows the loss in each iteration and Figure 8.9b shows the moving average for 100 iterations. That is the average of the 100 latest iterations. The loss oscillates around 20 and does not decrease. Figure 8.10 displays the ground truth and network output on two images from the Caltech pedestrian dataset. Overall it seems like the network output is higher in regions marked as *pedestrian* but values are rather high everywhere in the image.

Table 8.5 shows the average time for one forward pass, one backward pass and 1000 full passes in CPU and GPU mode. One forward pass of SqueezeNet plus the heatmap layer takes 2.21 times longer time in CPU mode and 2.54 times longer time in GPU mode than one forward pass of only SqueezeNet version 1.1 (see table 8.2).



(a) (b)

Figure 8.9: Loss during training on 10 000 randomly sampled images from the Caltech pedestrian dataset. (a) is the actual loss and (b) is the moving average for 100 iterations.

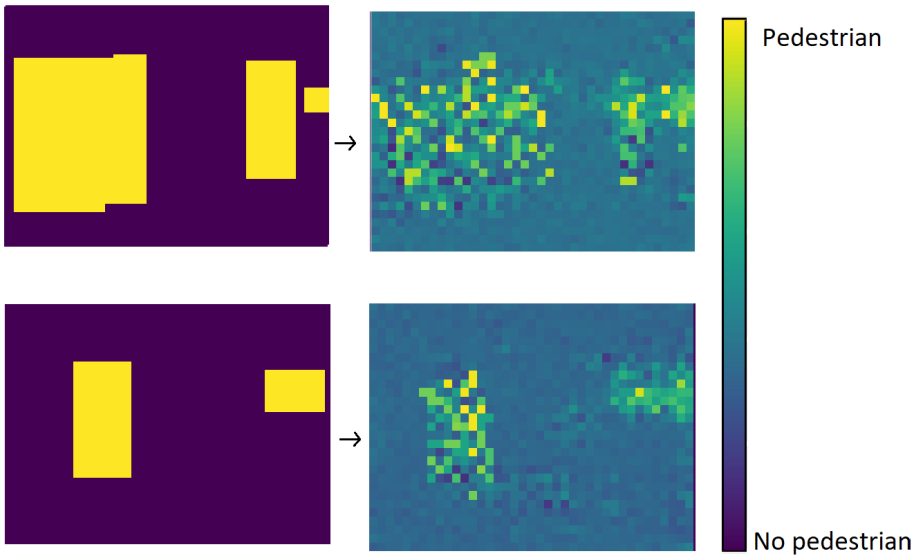


Figure 8.10: Ground truth and network prediction two images from the Caltech pedestrian dataset.

Table 8.5: Average time for one forward, one backward pass and 1000 full passes of the heatmap layer implementation

Mode	Forward pass[ms]	Backward pass [ms]	1000 full passes
CPU	4223.76	5403.23	2h 40min 27s
GPU	171.931	364.250	8min 29s

Chapter 9

Discussion

In this chapter the outcome of the project is discussed. In section 9.1 the findings from the literature study are handled and in section 9.2 discusses the design and implementation.

9.1 Literature study

Since the resurrection of ConvNets in 2012 they have dominated the field of image classification and object detection. In the early years the focus was mainly on improving accuracy which implied increased model sizes and increased requirements of compute power. This development continues and models of today even achieve better accuracy than humans. Lately there has also been a trend in making the existing state of the art models smaller and more effective. It seems like primary goal with ConvNets during the early years was to achieve as high accuracy as possible, no matter what model sizes were required. When the accuracy became outstanding high the focus shifted to make the models smaller while keeping the same accuracy. This development has been possible both due to the increased amount of available data and more powerful hardware but also due to improved algorithms. From the literature study it is found that the most computationally heavy parts of the network are the convolutional and fully connected layers. By reducing the number of or remove all the fully connected layers and chose filter sizes wisely these heavy computations could be reduced drastically which is promising for embedded applications.

The field of ConvNets for pedestrian detection is widely researched, and many approaches have been proposed. Pedestrian detection, which is a form of the more generic term object detection, meets high challenges both when it comes to the detection itself and when it comes to the implementation in real-world applications. Many challenges with the detection have been addressed with several different solutions for inter alia scale variations, occlusion, illumination etcetera. It seems like the most commonly used approach is the R-CNN based pipeline but since the region proposals often are time consuming this approach can be hard to implement in real-time. Faster R-CNN solves the time issue to some extent but with increased model sizes as a side effect, which is undesired in embedded systems. Another approach is to skip the region proposal step and introduce a heatmap layer as the final layer in the network. From the heatmap layer information about where pedestrians are located can be extracted. I believe this approach is promising in the perspective of implementation in embedded systems. The end to end system can be made smaller, less computationally heavy and thus faster even if there are still challenges in making this approach as accurate as state of the art faster R-CNN models.

There are both several advantages and disadvantages with FPGAs for ConvNet acceleration. My thoughts are that FPGAs will probably not replace GPUs for ConvNet acceleration due to the GPU capacity of performing floating point operations and regular parallelised computations. These are good characteristics especially for the training phase of the networks. The characteristics of an FPGA on the other hand suits well for the predictions which requires irregular parallelisation and when the system is to be implemented in an embedded system. The downside is mainly the limited memory resources, but this can be solved with network compression. Therefore FPGAs can constitute a complement to GPUs for embedded applications.

9.2 Design and implementation

During the classification on the INRIA person dataset running times of CPU and GPU execution was compared as well as the performance of SqueezeNet version 1.0 and 1.1 with weights initialised randomly and by transfer learning. Table 8.2 shows a comparison of the time needed to perform network training on the local CPU of a laptop and on the GPUs provided by AWS AMI Deep learning on Ubuntu. From the table it can bee seen that training the network on CPU is not feasible due to the extremely long training time. Therefore, it was decided to only use AWS for the rest of the implementation.

During training on the INRIA person dataset the loss decreases and the accuracy increases in a slower pace when the weights are initialised randomly than by transfer learning both for SqueezeNet version 1.0 and 1.1 and that the accuracy reaches one in both cases. This points out that features learned in the pre-training of the network are valuable for person classification as well, even if ImageNet which the networks are pre trained on, does not contain any humans. The reason that the accuracy reaches such high levels for both cases probably depends on that the dataset is relatively simple. There is either a pedestrian or not. The 6-fold cross validation shows the same results. Both networks are able to classify images in all testing sets correctly. Therefore, it was decided to validate the design further with a validation set that was not a part of the INRIA person dataset. Since the networks manage to classify all images in the validation set correctly as well the networks are either able to classify whether an image is of a pedestrian or not with extremely high accuracy or to simple images are only tested.

The initial heatmap layer implementation was tested on a simple problem just to see if the design was working or not. Since the loss decreases and reaches zero it seems like the design of the layer works. The many iterations it takes can be explained by the small network, only consisting of one convolutional layer. Due to the good results the decision to use the heatmap layer for detection on the Caltech pedestrian dataset was taken.

For pedestrian detection the network was initially trained on 1000 iterations on 1000 images. Among these images a large amount contain no pedestrians or only few pixels with pedestrians. This favours predictions to be zero or close to zero and it can be seen in Figure 8.4 that the network adapts to this since the loss decreases rapidly. When images containing more pixels labeled *pedestrian* come the network still predicts *no pedestrian* and the loss increases slightly. Since only a few images of pedestrians appears and since the loss is relatively low the network never learns to detect pedestrians, at least not during the first 1000 iterations. These images were not shuffled but shuffling them would probably not change the network predictions since it would still be the same amount of pedestrians and it would be in favour to predict only zeros everywhere. It seems like the penalty for missing a pedestrian is too small.

In order to check whether the network is able to learn from the images in the Caltech dataset or not, it was trained on a very small amount of images. The network indeed learned to detect the pedestrians in this small dataset but is heavily overfitted and unable to generalise to other images. The purpose of this test was to determine

whether the reason for the earlier, bad results was due to bad sampling of the images or an error in the implementation. Since the network seemed to work it was determined to continue with the set-up but with a different sampled dataset. On this dataset the loss decreases from around 140 to 20 in few iterations but then it does not decrease any more. Looking at the output the network seems to find areas where pedestrians are located but without high accuracy and certainty. The detection is not as perfect as the classification. This can of course depend on a number of different reasons but differences in the datasets can be one explanation.

Comparing the images in the two used datasets can explain why the extremely high accuracy achieved on pedestrian classification on the INRIA person dataset did not help achieving as satisfying results on the pedestrian detection on the Caltech pedestrian dataset. Figure 9.1 shows four examples of images containing pedestrians from the INRIA person dataset. It is very clear that there is a pedestrian in the images and all pedestrians are approximately of the same size. Compared to images without pedestrians these images differ a lot. Figure 9.2 shows four example images from the Caltech pedestrian dataset, also containing pedestrians. The images differ a lot from each other and the pedestrians appear in a large variety of different scales, in different angles from the camera and in different poses. The images are large and the pedestrians only constitute a small amount of the total image and the pedestrians does not necessarily differ a lot from the background, making the Caltech pedestrian dataset much harder than the INRIA person dataset.

The purpose to use a heatmap layer instead of the standard approach with R-CNN was to increase the detection speed. It takes 2.5 times more time to do one forward pass in the detection network than in the classification network. Since the classification network needs to process all regions that are proposed from a region proposal algorithm, the detection network will be faster if there are three or more regions proposed by a region proposal algorithm. In environments with complex backgrounds, such as urban environments it is probable that there are at least three interesting regions in each image, making the detection network operating faster.

Figure 9.1: Example images from the INRIA person dataset.

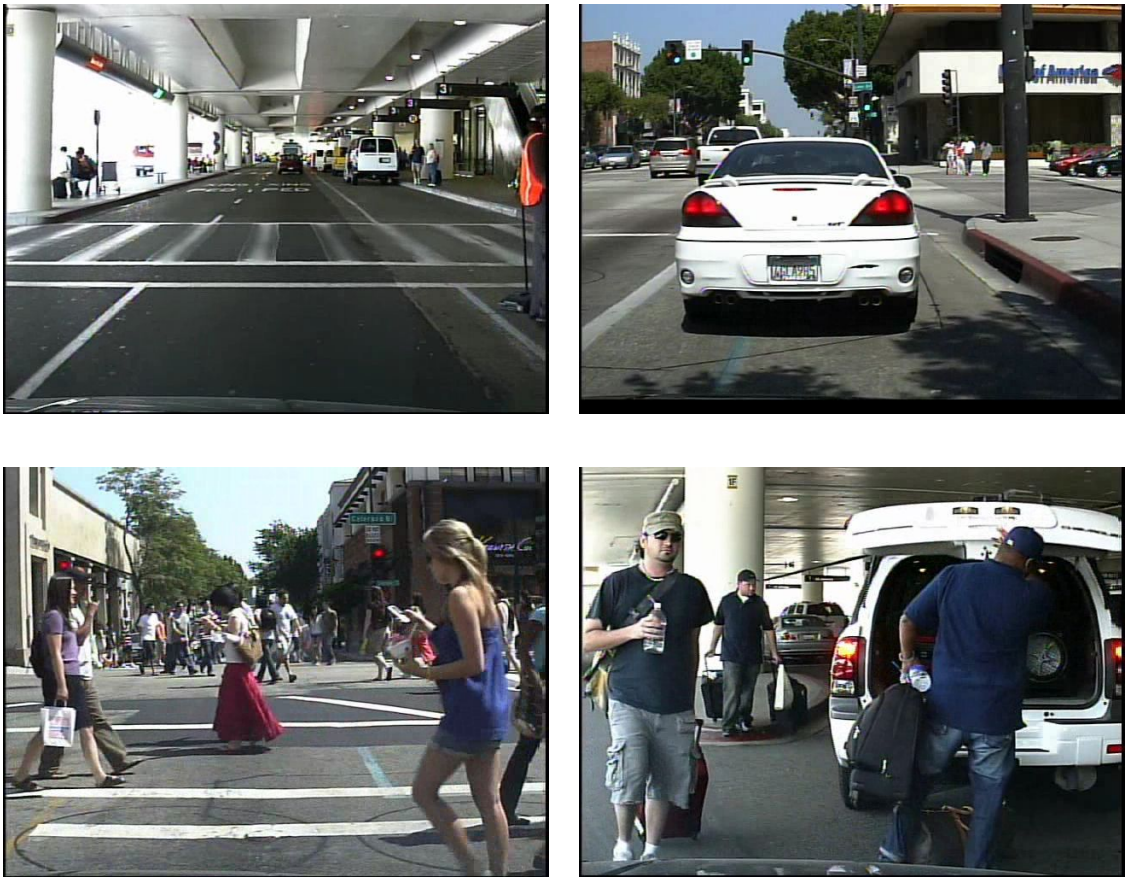


Figure 9.2: Example images from the Caltech pedestrian dataset.

Chapter 10

Conclusions and Future Work

This chapter presents the conclusions drawn from the project (sections 10.1) and what future work is suggested to do (section 10.2). In the Conclusion section the conclusions drawn from the literature study are presented first and then the conclusions drawn from the implementation and design are presented.

10.1 Conclusions

Overall conclusions drawn from the literature study are that the pedestrian detection problem is complex, mainly when it comes to the detection accuracy and speed. Many aspects of the problem have been investigated in previous works and many well-working solutions have been proposed. When it comes to pedestrian detection for autonomous vehicles, it is not enough to have a system that is able to do an accurate detection in all possible situations. The system must be possible to implement in an embedded systems as well which puts high demands on model size, required memory bandwidth and power consumption. For this purpose, FPGAs constitute a promising option due to their low power consumption and ability to perform parallelised computations.

The conclusions drawn from the first step of the implementation are that CPUs are not feasible for ConvNet computations and that the SqueezeNet network architecture works well for classifying images from the relatively easy INRIA person dataset. The accuracy reaches 1 both when using transfer learning and random weight initialisation but faster when using transfer learning.

Furthermore it is concluded that the heatmap layer implementation achieve promising results in terms of speed but there is still work to do in order to achieve stronger results in term of accuracy. Proposed improvements are described under section 10.2, Future work.

10.2 Future work

The heatmap layer implementation offers a fast approach to do the detection but does not achieve perfect results in terms of accuracy, yet. The first proposed solution to this is to use two or more parallel networks responsible for detecting pedestrians of different scales. All these networks should share the first layers but be different in the final layer. Secondly the training dataset could be created more wisely so that an even larger amount of the images contains a larger amount of pedestrians. Data augmentation such as mirroring can be used for this purpose. Furthermore the network should be penalised more when a pedestrian is not detected. Otherwise the network can "play safe" and never detect a pedestrian to a low cost i terms of loss.

As seen in this project the pedestrian detection problem is hard to solve, especially when aiming for implementation on embedded systems. In this project a small model size has been used with good results for pedestrian classification which is a much simpler problem than detection. Even though the used model size is small, it can be optimised for FPGAs further by pruning and quantisation while maintaining the same high accuracy.

This project has only focused on detection from single frames. No information from previous frames has been used. Full videos could be an option to use instead. Then information from previous frames together with other information such as vehicle speed could be used to predict where pedestrians are most likely to occur in the

current and future frames. Tracking by the use of optical flow is also an interesting further development. Once a pedestrian is detected it is tracked by the tracking algorithm. This could both decrease the number of computations performed by the detection network and increase the detection accuracy.

In this project the only used sensor has been an ordinary camera. In day light with good sight conditions an ordinary camera can be good for detection but since autonomous vehicles also need to function by night and in bad weather conditions, a further development could be to use other sensors such as IR-cameras and lidars in combination with the ordinary camera.

Acknowledgements

First of all I want to thank the other master thesis students at Alten; Rebecca Wikström, Marcus Olsson, Robin Moreno Rinding, Moaaj Bhana and Erik Sjölund for all support and giving discussions. I also want to thank my supervisor Detlef Scholle at Alten and my subject reviewer Carolina Wählby at Uppsala University for giving me the opportunity to write this thesis and for believing in me. Finally I want to thank Anton for endless patience and support.

Louise Augustsson
Stockholm 2018-06-08

Bibliography

[1] Gietelink O, Ploeg J, De Schutter B, Verhaegen M. Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations. Vehicle System Dynamics. 2006 july;44(9):569–590.

[2] Rawat W, Wang Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. Neural Computation. 2017 sep;29(9):2352–2449.

[3] Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems. 2012;25:1097–1105.

[4] Zeiler MD, Fergus R. Visualizing and Understanding Convolutional Networks. CoRR. 2013;abs/1311.2901.

[5] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. Proc International Conference on Learning Representations. 2014 sep;42.

[6] Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, et al. Going Deeper with Convolutions. CoRR. 2014;abs/1409.4842. Available from: `http://arxiv.org/abs/1409.4842`.

[7] Tomè D, Monti F, Baroffio L, Bondi L, Tagliasacchi M, Tubaro S. Deep Convolutional Neural Networks for pedestrian detection. Signal Processing: Image Communication. 2016;47:482 – 489.

[8] Hubel D, Wiesel T. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of physiology. 1962 jan;160(1):106–54.

[9] Nielsen MA. Neural Networks and Deep Learning. Determination Press; 2015.

[10] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998;86(11):2278–2324. IEEE.

[11] TECNALIA. About AMASS; 2018. Accessed: 2018-01-24. Available from: `https://www.amass-ecsel.eu`.

[12] SafeCop. Safe Cooperating Cyber-Physical Systems using Wireless Communication; 2018. Accessed: 2018-01-24. Available from: `http://www.safecop.eu/`.

[13] Z-turn Board (with Zynq-7020); 2018. Accessed: 2018-02-06. Available from: `https://www.xilinx.com/products/boards-and-kits/1-571ww1.html`.

[14] Leveson NG. Safety : Why , What , and How. ACM Computing Surveys (CSUR). 1986;18(2):125 – 163.

[15] Paul Gao, Russell Hensley, Andreas Zieĺke. A road map to the future for the auto industry | McKinsey & Company; 2014. Accessed: 2018-01-24. Available from: `https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/a-road-map-to-the-future-for-the-auto-industry`.

[16] Foot P. The Problem of Abortion and the Doctrine of the Double Effect. Oxford Review. 1967;5.

[17] Bonnefon JF, Shariff A, Rahwan I. The social dilemma of autonomous vehicles. Science. 2016 jun;352(6293):1573–1576.

[18] Levin S, Wong JC. Self-driving Uber kills Arizona woman in first fatal crash involving pedestrian . The Guardian. 2018;Accessed: 2018-03-20. Available from: `https://www.theguardian.com/technology/2018/mar/19/uber-self-driving-car-kills-woman-arizona-tempe`.

[19] Creswell JW. Research Design: Qualitative, Quantitative, and Mixed Methods Approaches; 2014. Sage publication.

[20] Fukushima K. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. Biological Cybernetics. 1980;36:193–202.

[21] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV). 2015;115(3):211–252.

[22] Liu Z, Dou Y, Jiang J, Xu J, Li S, Zhou Y, et al. Throughput-Optimized FPGA Accelerator for Deep Convolutional Neural Networks. ACM Trans Reconfigurable Technol Syst. 2017 Jul;10(3):17:1–17:23.

[23] Li FF, Johnson J, Yeung S. cs231n: Convolutional Neural Networks for Visual Recognition; 2017. Stanford university. Available from: `http://cs231n.stanford.edu/`.

[24] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986 oct;323(6088):533–536.

[25] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014;15:1929–1958.

[26] Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov R. Improving neural networks by preventing co-adaptation of feature detectors. CoRR. 2012;abs/1207.0580.

[27] Iandola FN, Moskewicz MW, Ashraf K, Han S, Dally WJ, Keutzer K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. CoRR. 2016;abs/1602.07360.

[28] Verbickas R, Laganiere R, Laroche D, Zhu C, Xu X, Ors A. SqueezeMap: Fast Pedestrian Detection on a Low-Power Automotive Processor Using Efficient Convolutional Neural Networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW); 2017. p. 463–471.

[29] Sermanet P, Kavukcuoglu K, Chintala S, Lecun Y. Pedestrian Detection with Unsupervised Multi-stage Feature Learning. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition; 2013. p. 3626–3633.

[30] Ouyang W, Wang X. Joint Deep Learning for Pedestrian Detection. In: 2013 IEEE International Conference on Computer Vision; 2013. p. 2056–2063.

[31] Hosang J, Omran M, Benenson R, Schiele B. Taking a deeper look at pedestrians. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2015. p. 4073–4082.

[32] Tian Y, Luo P, Wang X, Tang X. Pedestrian detection aided by deep learning semantic tasks. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2015. p. 5079–5087.

[33] Fukui H, Yamashita T, Yamauchi Y, Fujiyoshi H, Murase H. Pedestrian detection based on deep convolutional neural network with ensemble inference network. In: 2015 IEEE Intelligent Vehicles Symposium (IV); 2015. p. 223–228.

[34] Angelova A, Krizhevsky A, Vanhoucke V, Ogale A, Ferguson D. Real-Time Pedestrian Detection with Deep Network Cascades. In: Xianghua X, Mark WJ, Gary T, editors. Proceedings of the British Machine Vision Conference (BMVC). BMVA Press; 2015. p. 32.1–32.12.

[35] Li J, Liang X, Shen S, Xu T, Yan S. Scale-aware Fast R-CNN for Pedestrian Detection. CoRR. 2015;abs/1510.08160.

[36] Uijlings JRR, van de Sande KEA, Gevers T, Smeulders AWM. Selective Search for Object Recognition. International Journal of Computer Vision. 2013;104(2):154–171.

[37] Dollár P, Appel R, Belongie S, Perona P. Fast Feature Pyramids for Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2014 Aug;36(8):1532–1545.

[38] Viola P, Jones MJ, Snow D. Detecting pedestrians using patterns of motion and appearance. In: Proceedings Ninth IEEE International Conference on Computer Vision; 2003. p. 734–741 vol.2.

[39] Girshick RB. Fast R-CNN. CoRR. 2015;abs/1504.08083.

[40] Ren S, He K, Girshick RB, Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. CoRR. 2015;abs/1506.01497.

[41] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 1; 2005. p. 886–893 vol. 1.

[42] Felzenszwalb PF, Girshick RB, McAllester D, Ramanan D. Object Detection with Discriminatively Trained Part-Based Models. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010 Sept;32(9):1627–1645.

[43] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. In: CVPR09; 2009. .

[44] Nam W, Dollár P, Han JH. Local Decorrelation For Improved Detection. CoRR. 2014;abs/1406.1134.

[45] Dong P, Wang W. Better region proposals for pedestrian detection with R-CNN. In: 2016 Visual Communications and Image Processing (VCIP); 2016. p. 1–4.

[46] Zhang H, Du Y, Ning S, Zhang Y, Yang S, Du C. Pedestrian Detection Method Based on Faster R-CNN. In: 2017 13th International Conference on Computational Intelligence and Security (CIS); 2017. p. 427–430.

[47] Byeon YH, Kwak KC. A Performance Comparison of Pedestrian Detection Using Faster RCNN and ACF. In: 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI); 2017. p. 858–863.

[48] Guo A, Yin B, Zhang J, Yao J. Pedestrian detection via multi-scale feature fusion convolutional neural network. In: 2017 Chinese Automation Congress (CAC); 2017. p. 1364–1368.

[49] Ghosh S, Amon P, Hutter A, Kaup A. Detecting closely spaced and occluded pedestrians using specialized deep models for counting. In: 2017 IEEE Visual Communications and Image Processing (VCIP); 2017. p. 1–4.

[50] Ghosh S, Amon P, Hutter A, Kaup A. Reliable pedestrian detection using a deep neural network trained on pedestrian counts. In: 2017 IEEE International Conference on Image Processing (ICIP); 2017. p. 685–689.

[51] Cao C, Wang Y, Kato J, Zhang G, Mase K. Solving Occlusion Problem in Pedestrian Detection by Constructing Discriminative Part Layers. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV); 2017. p. 91–99.

[52] Hou YL, Song Y, Hao X, Shen Y, Qian M. Multispectral pedestrian detection based on deep convolutional neural networks. In: 2017 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC); 2017. p. 1–4.

[53] Choi H, Kim S, Park K, Sohn K. Multi-spectral pedestrian detection based on accumulated object proposal with fully convolutional networks. In: 2016 23rd International Conference on Pattern Recognition (ICPR); 2016. p. 621–626.

[54] König D, Adam M, Jarvers C, Layher G, Neumann H, Teutsch M. Fully Convolutional Region Proposal Networks for Multispectral Person Detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW); 2017. p. 243–250.

[55] Kruthiventi SSS, Sahay P, Biswal R. Low-light pedestrian detection from RGB images using multi-modal knowledge distillation. In: 2017 IEEE International Conference on Image Processing (ICIP); 2017. p. 4207–4211.

[56] Guo K, Sui L, Qiu J, Yu J, Wang J, Yao S, et al. Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2018 Jan;37(1):35–47.

[57] Nurvitadhi E, Venkatesh G, Sim J, Marr D, Huang R, Ong Gee Hock J, et al. Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks? In: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. FPGA '17. New York, NY, USA: ACM; 2017. p. 5–14.

[58] Lacey G, Taylor GW, Areibi S. Deep Learning on FPGAs: Past, Present, and Future. CoRR. 2016;abs/1602.04283.

[59] Moore A, Wiley J. FPGAs For Dummies ® , 2nd Intel ® Special Edition. New Jersey: John Wiley & Sons, Inc.; 2017.

[60] Christensson P. FPGA Definition; 2015. Accessed: 2018-02-20. techterms.com. Available from: `https://techterms.com/definition/fpga`.

[61] FPGA Acceleration of Convolutional Neural Networks - Nallatech; 2017. Accessed: 2018-03-07. Available from: `http://www.nallatech.com/fpga-acceleration-convolutional-neural-networks/`.

[62] Qiu J, Wang J, Yao S, Guo K, Li B, Zhou E, et al. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. 2016;34.

[63] Han S, Mao H, Dally WJ. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. CoRR. 2015;abs/1510.00149.

[64] Venkatesh G, Nurvitadhi E, Marr D. Accelerating Deep Convolutional Networks using low-precision and sparsity. CoRR. 2016;abs/1610.00324.

[65] Rastegari M, Ordonez V, Redmon J, Farhadi A. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. CoRR. 2016;abs/1603.05279.

[66] Dollár P, Wojek C, Schiele B, Perona P. Pedestrian Detection: An Evaluation of the State of the Art. PAMI. 2012;34.

[67] Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, et al. Caffe: Convolutional Architecture for Fast Feature Embedding. arXiv preprint arXiv:14085093. 2014;37.

[68] NVIDIA cuDNN -GPU Accelerated Deep Learning; 2018. Accessed: 2018-04-20. Available from: `https://developer.nvidia.com/cudnn`.

[69] SDSoC Development Environment; 2018. Accessed: 2018-04-20. Available from: `https://www.xilinx.com/products/design-tools/software-zone/sdsoc.html`.

[70] Amazon Web Services; 2018. Accessed: 2018-04-20. Available from: `https://aws.amazon.com/`.

[71] Lonvick CM, Ylonen T. The Secure Shell (SSH) Protocol Architecture. RFC Editor; 2006. RFC 4251. Available from: `https://rfc-editor.org/rfc/rfc4251.txt`.