BACHELORTHESIS

Using FCNNs for Goalpost Detection in the RoboCup Humanoid Soccer Domain

vorgelegt von

Jonas Hagge

MIN-Fakultät

Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Studiengang: Informatik

Matrikelnummer: 6809166

Abgabedatum: 25.06.2019

Erstgutachter: Dr. Norman Hendrich

Zweitgutachter: Marc Bestmann

Abstract

In this thesis, an existing fully convolutional neural network architecture that was able to detect one object class was extended for the detection of two classes. The neural network was used for the detection of the ball and was extended to also be able to detect goalposts in the RoboCup Soccer domain. Conventional methods were not able to accurately detect goalposts. It is important to know the position of the goalposts because this information can help in the self-localization of the robot. Additionally, it helps ensuring the robot kicks inside of the goal instead of next to it. The results show that the architecture can be expanded to output multiple heatmaps with a small trade-off in detection accuracy. The impact on runtime performance is negligible.

Zusammenfassung

In dieser Arbeit wurde eine Architektur eines fully convolutional neural networks, das vorher eine Klasse von Objekten erkennen konnte, erweitert für die Erkennung von zwei Klassen. Das neuronale Netz wurde genutzt für die Erkennung des Balles und wurde erweitert um außerdem Torpfosten finden zu können im Kontext des RoboCup Soccers. Konventionelle Methoden konnten nicht ausreichend genau Torpfosten finden. Es ist wichtig die Position der Torpfosten zu finden, weil diese Information helfen kann in der Selbstlokalisierung des Roboters. Außerdem hilft es sicherzustellen, dass der Roboter in das Tor trifft und nicht daneben schießt. Die Ergebnisse haben gezeigt, dass die Architektur erweitert werden kann, um mehrere Heatmaps auszugeben, mit einem geringen Einfluss auf die Erkennungsgenauigkeit. Der Einfluss auf die Laufzeit ist vernachlässigbar.

Contents

List of Figures

List of Tables

1. Introduction

Detecting goalposts is essential for the ability of a robot to play soccer. However, detecting goalposts using conventional methods is not very accurate.
Sec. 1.1 explains why the detection of goalposts is important and why a fully convolutional neural network will be used.
The goal of this thesis is explained in sec. 1.2.

1.1. Motivation

In recent years the improvements of deep learning have significantly helped the development of artificial intelligence systems. New ideas and concepts have led to breakthroughs in the field. This coupled with the availability of significant amounts of data has led to massive improvements. In the computer vision domain convolutional neural networks have led to significant progress. These methods are already in use by the Hamburg Bit-Bots to detect the ball on the field. The proposed methods in this thesis are primarily developed for use in the systems of the RoboCup team Hamburg Bit-Bots.

However, in the RoboCup Soccer domain it is not only essential to find the ball, but also being able to find goalposts on the field. Goalpost detection is important because it helps the robot to self-localize. This is important because odometry is not accurate enough for the robot to be localized on the field. By having an accurate self-localization, the robot is better able to decide where to kick the ball to score a goal. This helps with team play because the robots can communicate their position to the other robots. Based on this information the robots could e.g. pass and thereby get past a defending robot. Finding goalposts can also help decide in which direction the robot is currently looking. In between the goalposts, a goalkeeper of the defending team is likely to be present. Thus if two goalposts are detected and a robot of either team is also detected, the direction of the robot can most likely be determined correctly. Also, lines are currently detected in the Hamburg Bit-Bots vision pipeline by searching for white spots in the image. If a goalpost is detected, it can be ruled out as a possible line. By detecting the goalposts precisely the robot can decide to kick in a direction further away from the middle of the goal. If a goalkeeper has been detected, the corner in the goal further away from the goalkeeper can be chosen to kick at, because the goalkeeper is less likely to reach the ball in time.

The rules of RoboCup Soccer heavily restrict the allowed sensors [1] to ones that are human-like. Therefore, only the camera sensor could meaningfully help in finding the goalposts. Thus computer vision algorithms have to be relied upon. An example of a difficult situation is shown in fig. 1.1. This situation is very hard to solve with conventional methods. In this example, the neural network also detected a false positive. In other examples, the neural network presented in sec. 4.7 did however not detect false positives where a conventional algorithm would likely not be able to discern a true positive from a false positive. Fully Convolutional Neural Networks (FCNNs) have already been shown to work better in specific environments than neural networks which were state of the art before their introduction [LSD15].

1.2. Thesis Goal

The goal of the thesis was to evaluate if the architecture proposed by Speck et al. [SBB18] is able to also detect goalposts additionally to the ball without significantly increasing the runtime.

The network by Speck et al. only detected balls, however, this is not enough for the domain of RoboCup Soccer. The detection for the ball works reliably in real time on the limited hardware available on a mobile humanoid robot. That is why it was chosen as the one to evaluate.

The trained neural network should be able to reliably detect the goalposts on one output channel and the ball on another output channel while not sacrificing significant computation time.

1.3. Thesis Outline

The fundamentals that are necessary to understand the following sections are given in chapter 2.
In chapter 3 related approaches are explained. First, the general approaches in computer vision using neural networks and then the approaches for object detection in the RoboCup Soccer domain.
The experiments which were conducted in this thesis and why the experiments were done this way is explained in chapter 4.
How well the approaches worked is evaluated in chapter 5.
The methods applied in this thesis are discussed in chapter 6.
The thesis is concluded and possible future work is discussed in chapter 7.

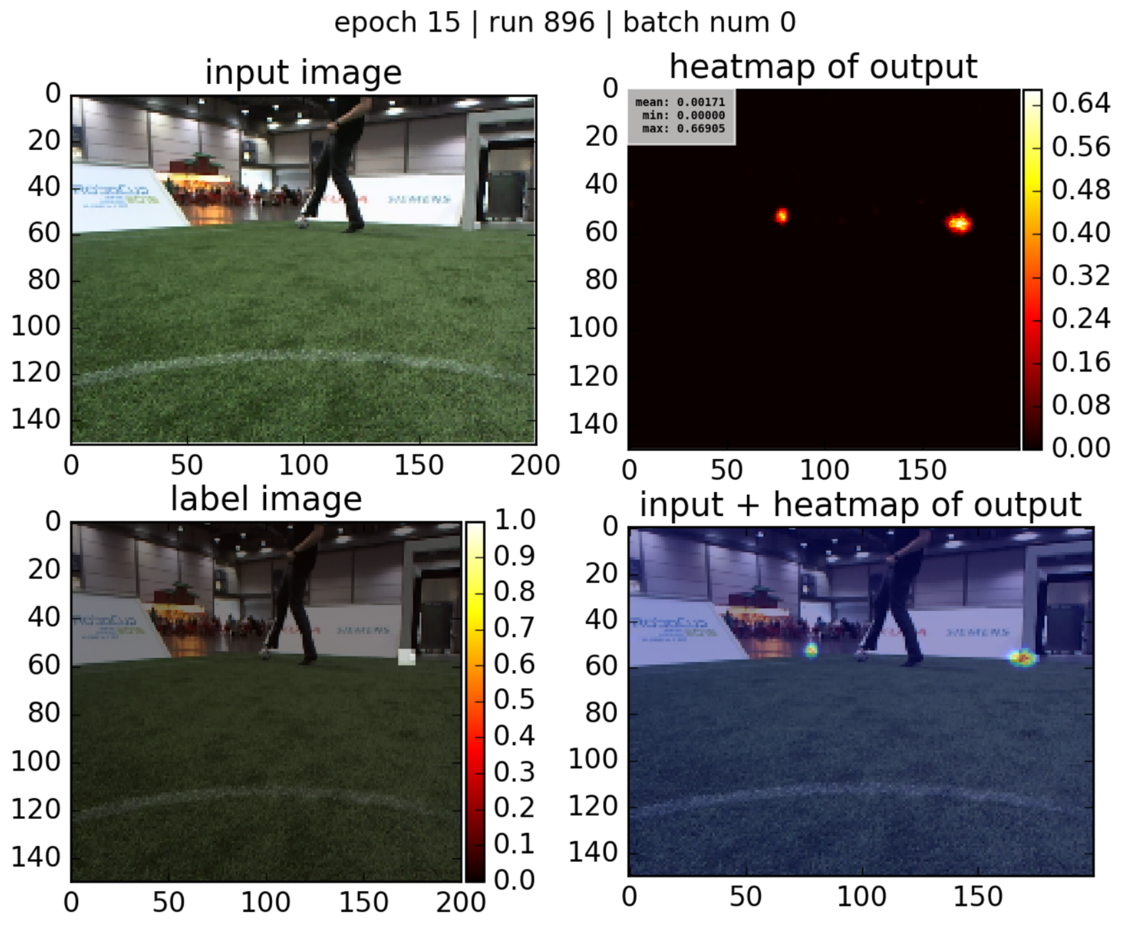

Figure 1.1.: Input image from [2]. This image from the dataset shows a difficult situation that happens in the RoboCup domain. The **top left** image shows the input to the neural network. The **top right** shows the output of the fully convolutional neural network, a heatmap. To the right of the image is a scale, because the scale is normalized. In this example, the highest activation is 0.6, but the highest activation is still white. The **bottom left** image shows the ground truth that was manually labeled which was cropped to learn only the bottom part of the label as explained in sec. 4.7. On the **bottom right** is the output of the neural network overlaid on the input image. The floor of the exhibition hall has a white spot which is vertical and thus looks very similar to a goalpost. The white spot is in the image space right above the green grass which does not allow distinguishing from a goalpost by requiring a field boundary right below it (either the network learning a goalpost is on grass or post-processing to require the goalpost to intersect with the field boundary).

2. Fundamentals

In this chapter, the fundamentals needed for the following chapters are presented. Sec. 2.1 introduces the RoboCup and presents the leagues of RoboCup Soccer.
Neural Networks are explained in sec. 2.2. Convolutional neural networks, a specific kind of neural network often used in computer vision are explained in 2.3. Fully convolutional neural networks, convolutional neural networks with no fully connected layer, are explained in 2.3.1.
The available computational platform on the mobile robots used by the Hamburg Bit-Bots and which parts of the robot are available for computer vision are described in sec. 2.4.
The ImageTagger which was the source of the training data is explained in sec. 2.5.
Since the computer vision on the robots is only able to find objects in the image space and not the relative position to the robot, the transformer is used. The transformer is explained in sec. 2.6.

2.1. RoboCup

RoboCup is an initiative which was founded in 1996 with the goal of promoting the development of artificial intelligence. The declared goal is to beat the champions of the human soccer world championship in 2050 with humanoid robots [3].

Multiple leagues have been founded in the RoboCup soccer domain. A Small Size League [4] with non humanoid robots, which have a camera on top of the playing field and a standardized vision system, which is computed on computers not on the field and then wirelessly communicated to the robots. The behavior of the robots is also not computed on the robots themselves, but from outside of the field of play. The main goal of this league is the coordination of multiple agents on the field while being in a highly dynamic environment.

The Middle Size League [5] requires the robots to have their own sensors on board. The robots also do not have to be humanoid. The computation has to be done on the robot and the ball in play is a regular FIFA soccer ball. The teams are free to design their own hardware with the only limit being a maximum height and weight. The league focuses on the hardware design of the robots, controlling these robots and also the coordination of multiple robots.

The Standard Platform League [6] requires all teams to use the same robot. Thus the teams are not able to gain any advantage via a better hardware platform. The software is the important part of this league and is significantly easier to compare from team to team, while still working with a real robot.

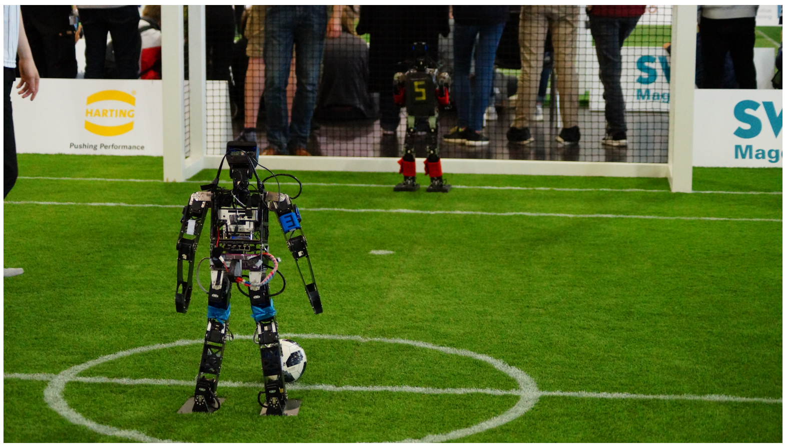

Figure 2.1.: Image used with permission by the owner. In this image the **blue** robot from the WF Wolves plays against the **red** robot of the Hamburg Bit-Bots. The game is played on a football pitch of 9 m$\times$6 m size.

Another league is the RoboCup Soccer Humanoid league [7]. This league is the one the fully convolutional neural network will be applied to since the Hamburg Bit-Bots play in the Humanoid league. There are different leagues in the Humanoid league with the main difference being the allowed height and weight of the robot.

The rules in this league [1] specify a field size of 9 by 6 meters. This is the distance the objects should optimally be detectable in. The ball as defined by the rules has to be "spherical[,] made of leather or another suitable material [and] FIFA size 1 for KidSize[...]"[1].

The Goal will be placed in the middle of the goal line (the 6-meter line of the field). The distance between the two goalposts is 2.6 m, the lowest point of the crossbar is 1.8 m and the goalposts do not extend higher than the crossbar. The goalposts can be "square, rectangular, round or elliptical in shape"[1] and have to have the same width and depth which must "not exceed 12 cm"[1]. The color for the goalposts as well as the crossbar is white.

2.2. Neural Networks

Artificial neural networks are networks made up of nodes that have been loosely inspired by how the human brain works. The goal hereby is not to copy the neural networks of a human but to use concepts that work well and apply them to different tasks. An example of a neural network is visualized in 2.2.

One such task can be computer vision, e.g. detecting and classifying objects in an image. An artificial neural network is made up of multiple layers [8]. The first layer is the input layer, the layer that is fed the information available to be analyzed. The last layer is the output layer which contains the output of the neural network. In between are layers,

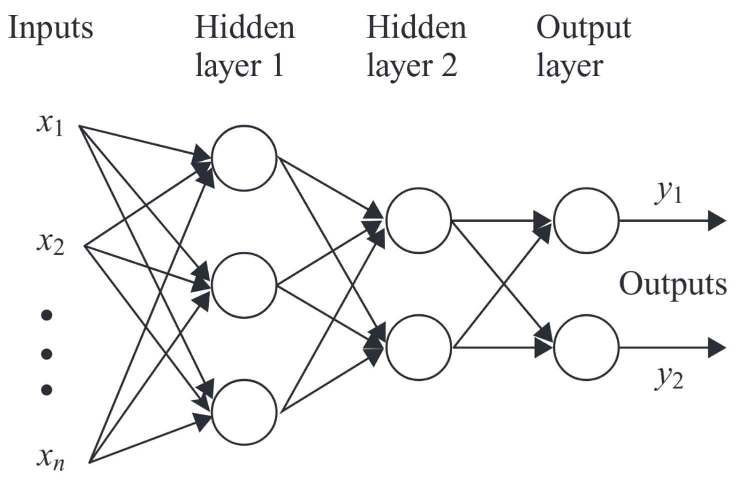

Figure 2.2.: Image from [Kan11]. On the **left** are the inputs which are fed into the neural network. Every input is connected to every node in the first hidden layer. On the **right** there are two output nodes.

called hidden layers, as their results do not contain either the input or the output and are thus usually not visible from outside the network. Nodes are connected to each other and in the case of a fully connected neural network, each node is connected to each node of the previous as well as the following layer. Each node has a weight attached to each connection it has to a node in the following layer. This weight determines how the value of the node influences the next node.

The activation function is also relevant for how the value of the node is propagated to the next layer. An activation function changes the value which is output from a node by for example returning 1 for every value larger than 0.5 (binary step). More sophisticated activation functions also exist. For example, the ReLU function which is the one that was used by Speck et al. [SBB18] and also the one the networks proposed in this thesis use. The ReLU function [GBB11] has as output zero for every value below zero and the input value for every value larger than zero. An activation function is useful because a nonlinear function makes it possible for a neural network to solve nontrivial problems while requiring fewer nodes [9].

Dropout [SHK$^+$14] is a method that randomly selects nodes as well as connections of nodes that are not used in the training of a neural network. Different nodes and connections are chosen for different training steps. This method reduces the problem of overfitting by preventing nodes from relying on the output of another node. This allows the network to e.g. in the computer vision context to learn a more general representation of the object rather than relying on the cases represented in the training set.

To improve the output of the neural network, methods such as Adam optimizer [KB14] can be used. It is used in the training of the neural networks in this thesis. The Adam optimizer is the method of how the weights are changed in the training. First, a forward pass is made through the network which means giving the neural network an input and the neural network will compute some output. This output is the prediction of the neural network. During training, the ground truth is known and the weights of the neural network can be changed to estimate the ground truth better. This is done by calculating the gradient of the influence of the weights on the output and changing the weights accordingly.

For the calculation of how different the ground truth and the output of the neural network is, a loss is calculated. The loss calculation for the proposed neural networks is based on the squared difference between the ground truth and the output of the neural network.

Overfitting means fitting the parameters too much to the specific training data[Kan11]. This leads to a model that can accurately predict the output for the training data but becomes inaccurate for data that was not included in the training set.

To have less overfitting the training data can be split into training and test data sets [Kan11]. The training data set is used to learn to estimate the ground truth. The neural network also predicts the output for the test data set, but the calculated error rate is not used to optimize the weights of the neural network. Instead, the test data can be used to see how well the network is able to predict previously unseen data. If the calculated error for the test set starts to increase instead of decreasing, the training should stop to not decrease the ability of the network to predict correctly on previously unseen data.

2.3. Convolutional Neural Networks

This section is based on [GBC16]

Convolutional neural networks are used to process data which is grid-like. An image is a 2-dimensional grid with, in the case of RGB, 3 channels. A convolutional neural network is a neural network that has at least one convolutional layer.

A convolution is an operation that is applied to two functions that are calculated with real numbers. In the case of convolutional neural networks, one of these functions is the input while the other function is a kernel. The output of this calculation can be referred to as a feature map. The input is a multidimensional array of color values, the kernel is a multidimensional array of parameters that are changed by the learning algorithm.

Fig. 2.3 shows how a kernel applied to the input looks like with an example. Unlike fully connected layers a convolutional layer does not have a matrix defining the interaction for every value of the input and the output. The convolution is usually smaller than the input, which allows reusing the same kernel to detect the same features in different positions of the image. This is useful because e.g. lines can be detected this way using fewer parameters since the same kernel is applied to all parts of the image. This reduces the memory required for storing the network and also means the neural network requires fewer operations. This is called parameter sharing.

Convolutional layers are equivariant. This means if the input to the layer changes, the output to the layer will change in the same way. This is a useful property because

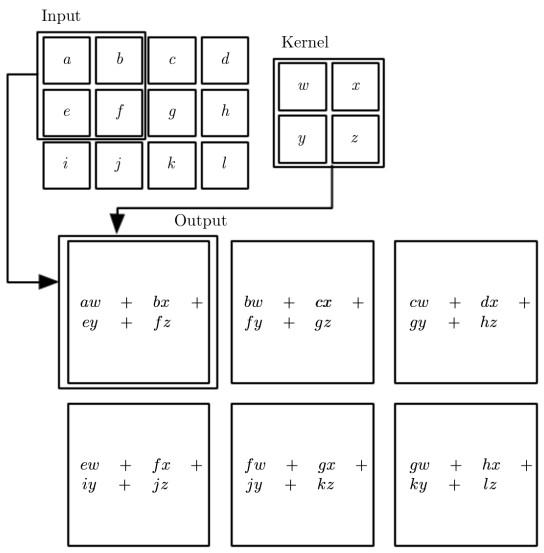

Figure 2.3.: Image from [GBC16]. In the **top left** the values a, b, e and f are multiplied with the values from the kernel. The values are added and produce one output value. This is repeated for the input positions for all possible kernel positions of the kernel being completely in the image.

it means the kernels can be used even if the input image size changes, because if the input changes in size the kernel can either be applied more or less often to the new input. This will create a different output, but the detection of the feature is not different, because the kernel is applied in the same way as to a different sized input. This also means the same filters can be used for a different context. For example, the first layer could be used to detect edges and the following filters can use the property of the detected edges.

Pooling is a function that takes multiple input values and returns only one. The max pooling function takes a rectangular region of the input and returns the maximum value of the input. This can be seen as a statistical summary of the region. This leads to invariance to translation of the input, because the feature can still be detected even though the exact position is not known due to the loss of accuracy by the pooling. Pooling can be used to reduce the amount of input parameters if e.g. the following layer requires a specific amount of parameters like a fully connected layer. With pooling and due to the equivariance property of the convolutional layers a convolutional neural network is able to handle different sized input images.

A stride can be used to skip over some possible positions of the kernel. In fig. 2.3 a stride of 1 is used, so the kernel is applied to the possible positions being completely inside the image. A stride of 2 would mean for this example after applying the kernel to the input values a, b, e and f the kernel would not be applied to b, c, f, and g, but rather would next be applied to c, d, g, and h.

It is possible to add padding to the input to make it wider or higher. This can be

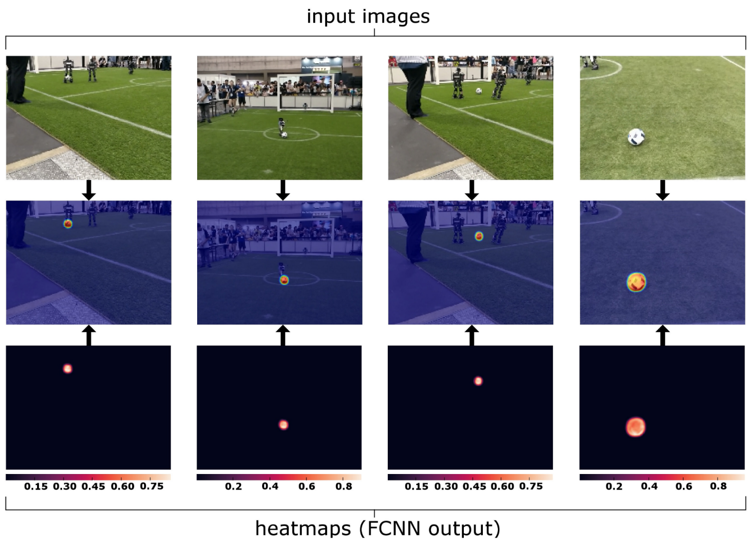

Figure 2.4.: Image from [SBB18]. At the **top** are the input images to the fully convolutional neural network. The **bottom** shows the output of the FCNN with high values at the positions where the network detected the ball. The images in the **middle** are the input images and the output heatmap laid on top of each other, showing the networks accurate detection of the ball.

for example a row of zeros, in which case it would be called zero padding. If this is not used, the output width will be smaller by one pixel less than the width of the filter. Without padding either the spatial extent of the network gets significantly smaller every layer or very small filters have to be used.

2.3.1. Fully Convolutional Neural Networks

This section is based on [LSD15]

Fully convolutional neural networks (FCNNs) are able to make pixelwise predictions. This means the output of an FCNN will not only detect which class is in the image or a bounding box encompassing the object, but a pixel-precise heatmap having values for each pixel for how likely it is for that pixel to be part of a specific class. This means an FCNN can not only do classification but also pixel-wise segmentation of the image. An example of this can be seen in fig. 2.4. This is accomplished completely inside of the neural network without the need for pre- or postprocessing. The FCNNs keep the property of convolutional neural networks of being translation invariant. They only rely on relative spatial coordinates. Due to how convolutional layers work, an FCNN is able to operate on input of any size and will have an output of corresponding spatial dimensions.

The fully connected layer included in most convolutional neural networks can be viewed as being a kernel which covers the entire input region of the previous layer.

If the fully connected layer is replaced with a kernel of the size of the input, the network will be an FCNN which will generate a heatmap corresponding to classes. If this method is applied to AlexNet [KSH12] this will lead to the network requiring a computation time of 5 times less than the amount of the original network.

Pooling is used to downsample the image like it is used in convolutional neural networks. In FCNNs there is also upsampling. This is done via a backwards convolution, also called deconvolution, which is basically just a reverse of the forwards and backwards pass of the convolution. This is a process that can be learned by the neural network.

To address the lost information by pooling and using a bigger stride, skip layers can be used which combine later layers with information from earlier layers to make use of previously available information in the image [RFB15]. This information helps the deconvolution process to then generate a more precise segmentation of the input.

By using a large number of feature maps in the upsampling process the network is able to provide context information to the higher resolution layers [RFB15]. This leads to a network that is almost symmetrical for both the downsampling process as well as the upsampling process.

The downsampling process part of the architecture can often look like a regular convolutional network [RFB15].

2.4. Computational Platform

The Wolfgang Platform used by the Hamburg Bit-Bots is a robot with 20 degrees of freedom [BBE$^+$19]. The camera used is a "high-definition USB Webcam" [10]. The rules allow the usage of two cameras which could be used for stereoscopic vision, however, no team makes use of this, because to the best of my knowledge the problem is the cameras have to be calibrated very precisely about the position of the other camera. In the RoboCup domain, robots will regularly fall, which leads to the cameras being not at the exact position they were calibrated in. Since a very small error in calibration can lead to a significant difference in the calculated relative position of the detected object, this approach is not used.

The robot has as computing units available an Intel NUC, an Nvidia Jetson TX2, and an Odroid XU-4 [BBE$^+$19]. Of these, the Nvidia Jetson TX2 is used for the vision part of the processing. The Nvidia Jetson TX2 provides a 256-core NVIDIA Pascal™ GPU [11], a Dual-Core NVIDIA Denver 2 64-Bit CPU [and] Quad-Core ARM® Cortex®-A57 MPCore [11] with 8GB 128-bit LPDDR4 Memory [11].

2.5. ImageTagger

For the training of the neural networks images and labels containing the position of the goalpost as well as the ball are required. The ImageTagger [FBH18] provides an open source platform for labeling images. It is collaborative and multiple teams in the

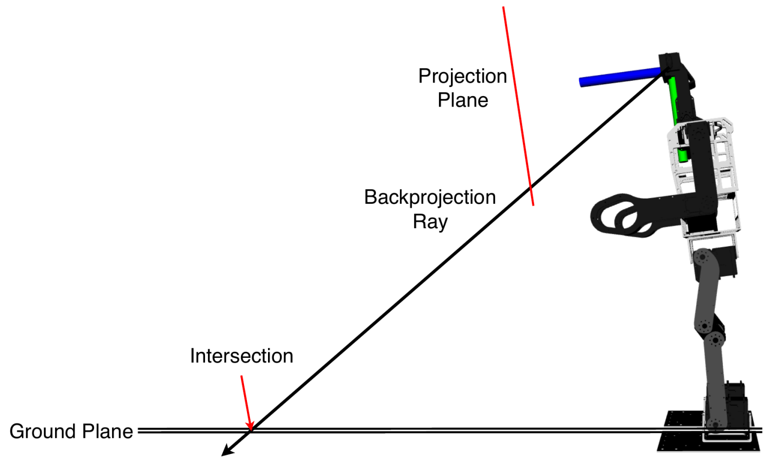

Figure 2.5.: Image from [Gü19]. The transformation calculation visualized. The projection plane is the image and a ray is calculated through the position in the image space of the object to find the intersection of the ray and the ground.

RoboCup have labeled images and uploaded new images from e.g. the perspective of their robots. The ImageTagger provided by the Hamburg Bit-Bots has 290.000 images of the RoboCup domain publicly available and 30000 labels of goalposts [2]. The labels for the goalposts were created by humans defining four points which together outline the goalpost. The balls were labeled with a bounding box encompassing the ball. Tens of thousands of labels have been created in the context of this bachelor thesis by the Hamburg Bit-Bots team.

The image collections are split into image sets. This way the images can be sorted by context of where the images were taken or for example which camera was used. It also makes it easy in the training process to only include some of the image sets.

The ImageTagger instance of the Hamburg Bit-Bots [2] is the source of the training and test sets used in this thesis.

2.6. Transformer

The objects are found in the image space and no depth information is available as explained in sec. 2.4. To interact with its environment the robot needs to know the relative position of the objects to it. This calculation is the task of the transformer. The transformation process is visualized in fig. 2.5. The projection plane is the image. A ray is calculated going through the projection plane to the ground. The ray has to be calculated because the rules prohibit active sensors. This ray starts from the position of the camera which is calculated from all the joint angles from the robots. This requires precise calibration of all the joints. From the distance, the ray takes from the position of the camera, the relative position of the object is calculated.

3. Related Work

Sec. 3.1 gives an overview of general approaches for computer vision, specifically computer vision with neural networks

The approaches for the task of detecting goalposts and other objects on the field in the RoboCup Soccer domain are explained in sec. 3.2.

3.1. General Approaches

There are many approaches available for object detection in images. The more conventional approaches consist of handcrafted filters which are very time consuming to develop. More recent approaches favor deep neural networks, more specifically convolutional neural networks. Convolutional neural networks have become very popular after AlexNet [KSH12] was very successful in the ImageNet challenge in 2012. It won the competition with only a 15% error rate on their top 5 guesses for which object is in the image. The second place had an error rate of 25%.

The VGG Net [SZ14] showed that increasing depth by using smaller filters improves the results of the network. The VGG Net used smaller 3x3 filters unlike e.g. AlexNet which used larger filters but fewer layers.

In medicine, the U-Net [RFB15] has been very successfully used for image segmentation. In this case not only image classification is needed, which the AlexNet already performed well for the ImageNet challenge, but also the image has to be segmented into several object regions. This means the location of different objects has to be detected. The Task given is an image of cells. The cells have to be segmented in the image and then they have to be classified. Another challenge in this domain is the very low amount of training images. Thus data augmentation was heavily used for the training of the U-Net.

YOLO [RDGF16] puts a grid of predefined size on the image. Then the architecture is split to fulfill two tasks. One of these tasks is to predict where an object is in the image. This part does not classify what the bounding box contains, it only finds most probable positions for objects. The classification is done in the other task that is run simultaneously. Each grid element is classified as one object. The bounding boxes passing a confidence threshold are then evaluated and a class is chosen for each bounding box. This approach is faster than competing approaches [RF17] while maintaining very high accuracy. Additionally, a model called Tiny YOLO has been published [12] which makes the trade-off of accuracy for speed of detection. This architecture would be the most suitable to achieve real-time detection on the limited hardware available on the robot.

3.2. Approaches in RoboCup

Some teams use comparably basic algorithms for computer vision. The team Electric Sheep uses color for candidate detection. Then the best ball candidate is chosen based on "roundness, colour profile, distance, size and distance from previously identified location" [BCBK$^+$19].

The current approach for goalpost detection of the Hamburg Bit-Bots is based on the detection of the field boundary [FBG$^+$19]. The field boundary is the edge of the field. For the detection of the field boundary, a lookup table will be created containing every possible RGB value. The color values which are part of the field are defined by a human prior to the game by selecting areas of green pixels that are part of the field. In the detection of the field boundary, the topmost pixels which have previously been defined by the human as green are searched in a column from e.g. top to bottom. This is done for a predefined value of columns. If something is in the way of the robot it will block the robot's line of sight to the edge of the field. This will leave a dent in the detected field boundary. The area above the dent is analyzed for color and if the amount of e.g. white reaches a threshold the object is assumed to be a goalpost. This approach is not very accurate and only accomplishes a mean Jaccard-Index of 0.183 [FBG$^+$19].

The team EagleBots.MX uses Haar-like features with a cascade classifier for object detection [LVMR19].

Other teams use YOLO [RDGF16] for detecting multiple objects including ball and goalposts [RAS$^+$17, HHM$^+$19]. Rudiawan et al. [RAS$^+$17] show the method is successful in detecting balls and goalposts in the RoboCup Soccer domain. For this method, only 4000 images were used for the ball and goalposts each. The used Jetson TX1 is able to detect the objects with 45 frames per second. However, by using YOLO, there is a trade-off as explained in sec. 4.2.

Another approach in the RoboCup Soccer league is to use Fully Convolutional Neural Networks. One such approach is presented in [vDS18], where a new architecture was developed to first find balls and later find goalposts and balls in the same network. To find goalposts as well as balls a new channel was added in the last decoding layer. For the goalposts, only the bottom part of the goalpost was used for the learning. The bottom part of the goalpost means, in this case, a circle with a 20-pixel radius where the goalpost touches the ground. Adding goalposts to the approach only lead to a small decline in accuracy of detecting balls and the computation time needed was also not significantly increased. This approach is similar to the one that is presented by this bachelor thesis, as it also uses a fully convolutional neural network to detect balls and goalposts. However, the architecture used by van Dijk et al. is a different one than the one that is presented in this thesis.

Another method employed in the RoboCup Soccer domain is the one presented for the robot Sweaty [SSW$^+$17]. The neural network is able to detect the ball, opponents, goalposts as well as field features (e.g. lines). This architecture takes about 11 ms (meaning about 91 frames per second) to process an image on the Sweaty robot while maintaining high accuracy. 2150 images were used for the training of the network as well as data augmentation on these images. The network was trained to learn the center of the object. Using dropout as part of the architecture led to a worse recall and a higher false detection rate. This architecture is able to detect most of the relevant objects on the field of the RoboCup Soccer domain while being accurate and fast. However, the approach also uses a GeForce-GTX-760 GPU on the robot, which is not feasible to use in a humanoid kid size robot and thus not applicable for this thesis.

Team Rhoban who won the 2018 world cup in the humanoid kid size league uses a classification approach for object detection [AGH+19]. For this approach regions of interest for the ball and goalposts on the image are detected in the image, by using the current state of the robot as well as “a kernel convolution on an Integral Image filter” [AGH+19]. After deciding on the regions of interest, they are classified by a convolutional neural network. With this approach, Rhoban is able to achieve about 97% accuracy for both balls and goalposts. This approach uses only a classifier and relies on a good region of interest selection. The approach of classifying regions of interest also means the regions might be too large and viable candidates could be filtered out before the convolutional neural network could classify them. To be able to get more precise detections of the objects, an approach that does not only classify regions of interests is preferable. As explained in sec. 4.2 an approach using a bounding box is not preferable for the detection.

The approach presented in [GHSG18] proposes a convolutional neural network to classify patches. It uses pretrained networks e.g. AlexNet [KSH12] which are then being retrained with images from the RoboCup domain. The patches are 50x50 pixels in size and chosen by generating binary images based on if the pixels are green (for the field) or white (for the ball). If one patch has more than 30 detected white pixels it is chosen to be a candidate for the ball. For the training, all patches are used, while during detection only the ball candidate patches are evaluated. A region of interest approach which generates bounding boxes is not preferable as explained in sec. 4.2.

The architecture presented by Speck et al. [SBB18] used 35327 images for training and 2177 images for evaluation. It is only used to detect the ball. There are two architectures presented, model one is less resource intensive, but also less precise. Model 2 requires more hardware but also gives better results. It is based on the architecture presented in [SSW+17] but uses a smaller input image of 150x200 size instead of the originally used 512x640. In this network, a dropout rate of 0.5 is applied to all but the first and last layer. Contrary to the results in [SSW+17] the dropout did improve the results of the network. The network was also evaluated on a negative dataset. This means no ball was in any of the pictures of the dataset. The network detected a ball in their specific dataset in only 1.04% of the images, showing it is unlikely to produce a false positive for their dataset. With a batch size of 1, the network took on average 0.049 seconds for the detection of the image. Since the network on the robot will work with a batch size of 1, 20 frames per second can be detected. Model 1 is mostly shown as a reference and for use with robots only having a CPU available for computation.

The proposed model 2 takes an input of 150×200 pixels with 3 channels. This is put into filters of size 3×3 followed by a batch normalization and max pool layers of

2×2 size. Additionally, some of the filters are used in later concat layers to have more information available of the original image which allows the network to use the original information of the image to generate a pixel-precise detection. The output is a heatmap of size 150×200 with one channel. It is visualized in fig. 4.4 on page 23.

Since the approach of the proposed model 2 has been shown to work well on the robots of the Hamburg Bit-Bots the approach in this thesis will build on the results of [SBB18].

4. Approach

This chapter explains which decisions were made in the experiments for this thesis.
In sec. 4.1 the format to download the labels of the data from the ImageTagger is presented.
Sec. 4.2 explains why an FCNN is advantageous to use over other approaches such as YOLO.
How the network was trained is explained in sec. 4.3.
An explanation of how the humans labeled the data with the position of the goalpost is given in sec. 4.4. The decision to use images where the object has been labeled as not in the image is explained in 4.5.
Using the architecture shown in fig. 4.4 a neural network is trained with a single output channel and with the full goalpost label as the ground truth. This is explained in sec. 4.6.
For using the same architecture but only learning the bottom part of the goalpost label an experiment is made in sec. 4.7.
An explanation of why using two output channels in one architecture instead of using two separate neural networks is given in sec. 4.8
The approach of learning the bottom part of the goalpost with two output channels of the neural network is explained in sec. 4.9. For this and the following experiment the architecture visualized in fig. 4.10 is used.
Sec. 4.10 shows how the approach of using two output layer for the neural network while training with the full goalpost label was done.

4.1. ImageTagger Export Format

The framework developed by Speck et al. [SBB18] previously used a custom export format. The format was then parsed and saved in a Python dictionary. This method does not allow for easy extendability since every change in the export format will also have to be changed in the parser. One such change could be also including goalpost labels in the output format. Since the export format from the ImageTagger is customizable, a new export format was created. The new export format is in the YAML [13] format. This format can be parsed by existing libraries such as PyYAML [14] and the result is saved in a Python dictionary. This method allows for easier extendability since the parsing method can stay the same. It also allows for easier collaboration since the YAML format is a standardized data-serialization language and is thus more likely to also be used by other teams that could provide images and labels.

The Python dictionary consists of two parts. The first part is simply the name of the image set. The second part consists of the labels for the images. For each image, if a label for it exists, the content of the label is saved. It consists of the image name, which is used as an identifier which is then paired with the filename of the image on the disk. Additionally the width and the height of the image, which is used to calculate a scaling factor, because the image in the neural network will be downscaled and thus the label also has to be downscaled. The amount of labels for this image is also saved as well as the label itself. The label is saved as a list of lists of x and y coordinates.

Listing 4.1: A section of an example of the generated YAML

```
set: bitbots-montreal-game02
images:

    montreal-game02_aa_000001.png:
        width: 800
        height: 600
        annotations:
            - {type: goalpost,
            inimage: true,
            vector:
                [                 [463,261],
                [469,20],
                [457,24],
                [449,264],

                ]}
          - {type: goalpost,
            inimage: true,
            vector:
                [                 [715,312],
                [742,4],
                [716,3],
                [690,309],

                ]}
[...]
```

4.2. FCNN instead of YOLO

YOLO is a very popular approach in the RoboCup domain for detecting multiple classes in an image. Also outside of the RoboCup domain, this approach is commonly used. The advantage of using an FCNN over YOLO is the pixel precise detection of the object in the image. This is significant for the transformation into relative space. A bounding box that detects the goalpost inaccurately in the vertical dimension will lead to a significant error. This is a problem since the localization relies on the position of the goalpost being accurately detected. If the view of the robot is tilted, which commonly happens while walking, a bounding box approach will also lead to a bounding box that is significantly bigger than the actual size of the goalpost. If the goalpost top part is e.g. in the top left of the image, while the bottom part is in the bottom right, an accurate detection would lead to a bounding box encompassing the entire image. This is however not a precise detection of the goalpost since the goalpost could only be a thin bar that does not fill the entire image. One possible solution for this would be to use the sensor data from the IMU included in the robot. This could help to have an image which is less tilted and would likely solve this problem.

The FCNN approach is able to make pixel precise detections which is preferable over a bounding box because it is more accurate. Additionally, the approach presented in [SBB18] has been shown to work on the platform used by the Hamburg Bit-Bots and has shown good results for ball detection. The approach of [vDS18] has shown their architecture is able to detect the ball as well as the goalpost by just adding another output channel. It is likely the architecture of Speck et al. can be modified in a similar way to be able to detect the ball as well as the goalpost.

4.3. Training

The training of the neural network was done on an Nvidia Titan X Pascal. The framework was adopted and extended from the one used by Speck et al. in [SBB18] which provides a parallel algorithm for the data loading process to use the resources of the GPU as efficiently as possible. The training took about an hour. The training used different image sets for the training process and testing purposes. No image of the test set was included in the training set and thus an indication of if the neural network started overfitting could be seen from if the loss on the test set was still falling or started increasing again. The loss on the test set did not fall any lower anymore after about 15 epochs and started increasing instead, thus the training was stopped after 15 epochs for most of the training experiments.

4.4. Labels

To start the supervised learning, labels of goalposts were needed. The goalpost was defined for the labeling as from the bottom of the goalpost where it is connected to the ground and up to the crossbar, but not including the crossbar. An example of this can be seen in fig. 4.1.

It had to be decided if all goalposts in an image should be labeled as a goalpost or if specific exceptions should be made. The problem with labeling all goalposts as such would be, if multiple fields are close together and the robot is able to see more than just the goalposts of its field. One example of this is seen in fig. 4.2. In this case, the goalposts are not visibly attached to the ground. This is one main visible difference to

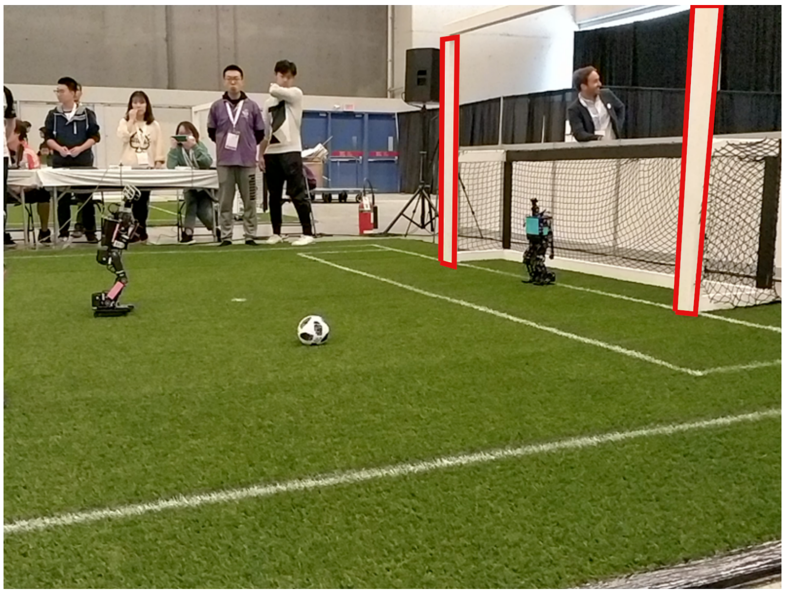

Figure 4.1.: Image from [2], edited to make label more visible. An example of manually labeled goalposts. The red outline encompasses the goalpost and will be used as ground truth for the training of the neural network.

e.g. pillars which are also often visible in the RoboCup soccer domain. The pillars will not be on the grass and thus with finding goalposts on the grass, goalposts on other fields can be ruled out and also other objects like the pillars. Thus it was chosen to not label goalposts on other fields where they are not visibly attached to the grass. Another example is shown in fig. 4.3, in this example it is not possible to rule out the goalposts on the other field by basing it on them being on grass. So for this case, it was chosen to label the goalposts on the other field too. It is not easily possible to rule out these goalposts as not relevant for the current field of play, just by the way they look. Additional features have to be taken into consideration, e.g. the game logic of ignoring anything where the field of play is interrupted from looking forward from the robot's point of view. This can easily be done in post-processing the output of the neural network. The vision pipeline used by the Hamburg Bit-Bots already searches a field boundary to not require image processing on parts of the image which are not relevant. If a small threshold is applied to this boundary because goalposts will be in large part above the boundary line, then the goalposts on other fields will be filtered out and the net possibly also learns goalposts features of goalposts only seen from a larger distance.

4.5. Using Negative Data

In the existing framework images which were labeled with an object "not in image" were skipped. This can be motivated because every image containing the object consists

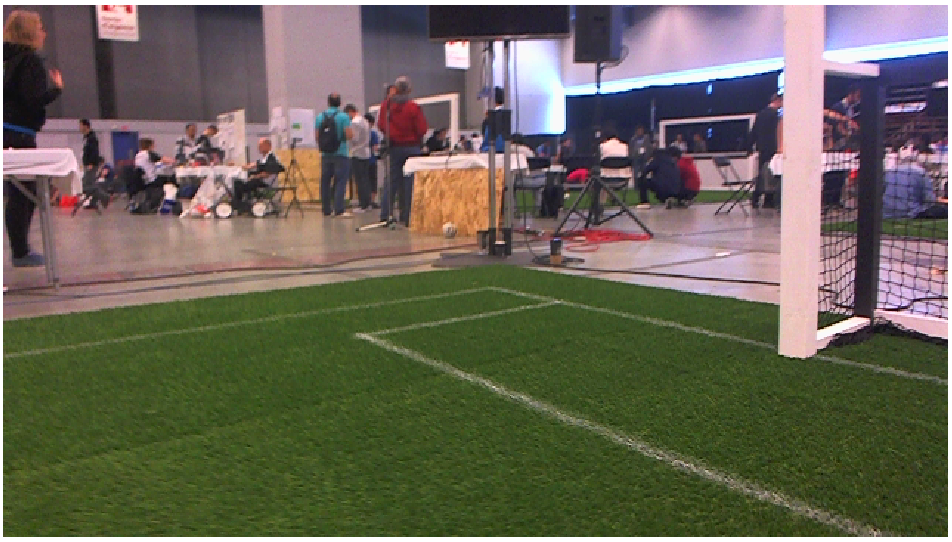

Figure 4.2.: One Image from the ImageTagger database [2] demonstrating goalposts positioned on other fields. Two fields are visible including their goalposts. This is a case in which it is difficult to determine which goalposts should manually labeled.

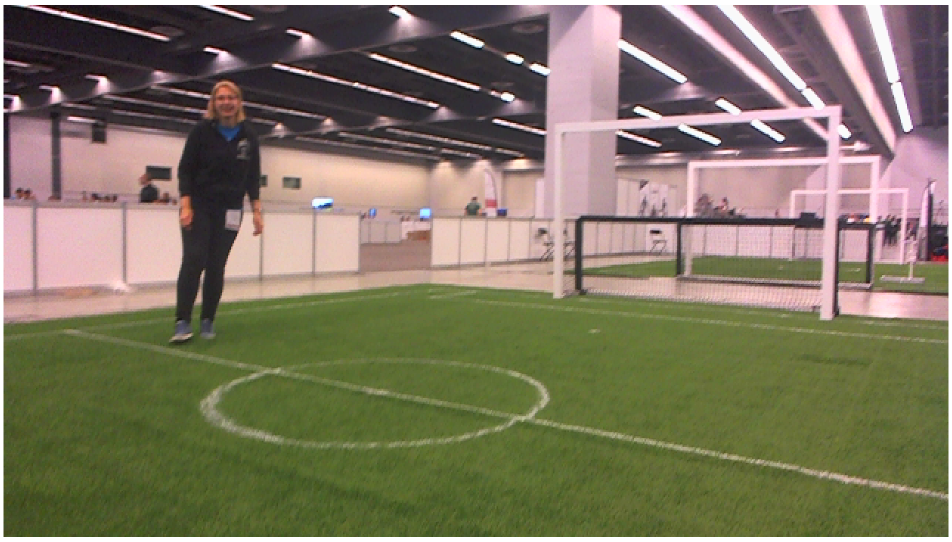

Figure 4.3.: Image from the ImageTagger [2] showing multiple goalposts on multiple fields. The field below each goalpost is also visible. Thus the decision to label in this case cannot be distinguished by the bottom of the goalpost being visible on the field.

mostly of parts that are not part of the object and thus all of this negative data is included in the training and can be used by the neural network to not have false positives. However, the team of the Hamburg Bit-Bots had a problem with false positives detected by the neural network for ball detection, because e.g. the arm of the robot was sometimes detected as a ball. Thus the decision was made to include negative data in the training process. If the experiments showed to have too many cases of having false negatives it could easily be changed back to not include negative data in the training process.

4.6. One Channel - Full Goalpost Label

The first approach of training goalposts used the full label which has been created as explained in sec. 4.4. The label contains the four coordinates of the label in the order in which they were created in the ImageTagger. This means the ground truth masks can be created by drawing an outline following the points given from the export from the ImageTagger and then filling the content of this outline with the value of 1.0 for each pixel inside of the label.

The architecture used for this experiment is the one proposed by Speck et al. [SBB18]. It is visualized in fig. 4.4.

This approach worked to find goalposts over a significant distance as shown in fig. 4.5. The detection, in this case, was also over a large distance of more than half the field and there were no significant activations in the output which means no false positives were present in the output in this example.

The detection also worked for goalposts closer to the robot as shown in fig. 4.6. This case is also made more difficult as the ball is right next to the goalpost which makes the surrounding area of the goalpost look different than most images present in the training set. This could explain why the activation is mostly present in the upper parts of the visible goalpost and there is a significantly smaller activation in the lower parts of the goalpost. However, this is a very important case for the robot as this situation is one which will likely happen during a regular game where the ball is very close to the goalpost. In this case, the robot should reliably find the goalpost to not kick the ball next to the goal but inside of the goal.

In fig. 4.7 two examples of false positives detected by this network are visible. Pillars like the one present in these two images are common in the RoboCup Soccer domain because games are played in exhibition halls where such pillars are common. Thus this is a significant problem. It is possible that the main feature the network learned was a white vertical line in front of an indiscriminate background because the backgrounds of the goalposts can be very different depending on e.g. the angle the goalpost is seen from as well as people standing behind the goal or a different position of the exhibition hall. Only a small part of the goalpost is connected to the field which will always look very similar because the field consists of grass and the white lines the goalpost is connected to.

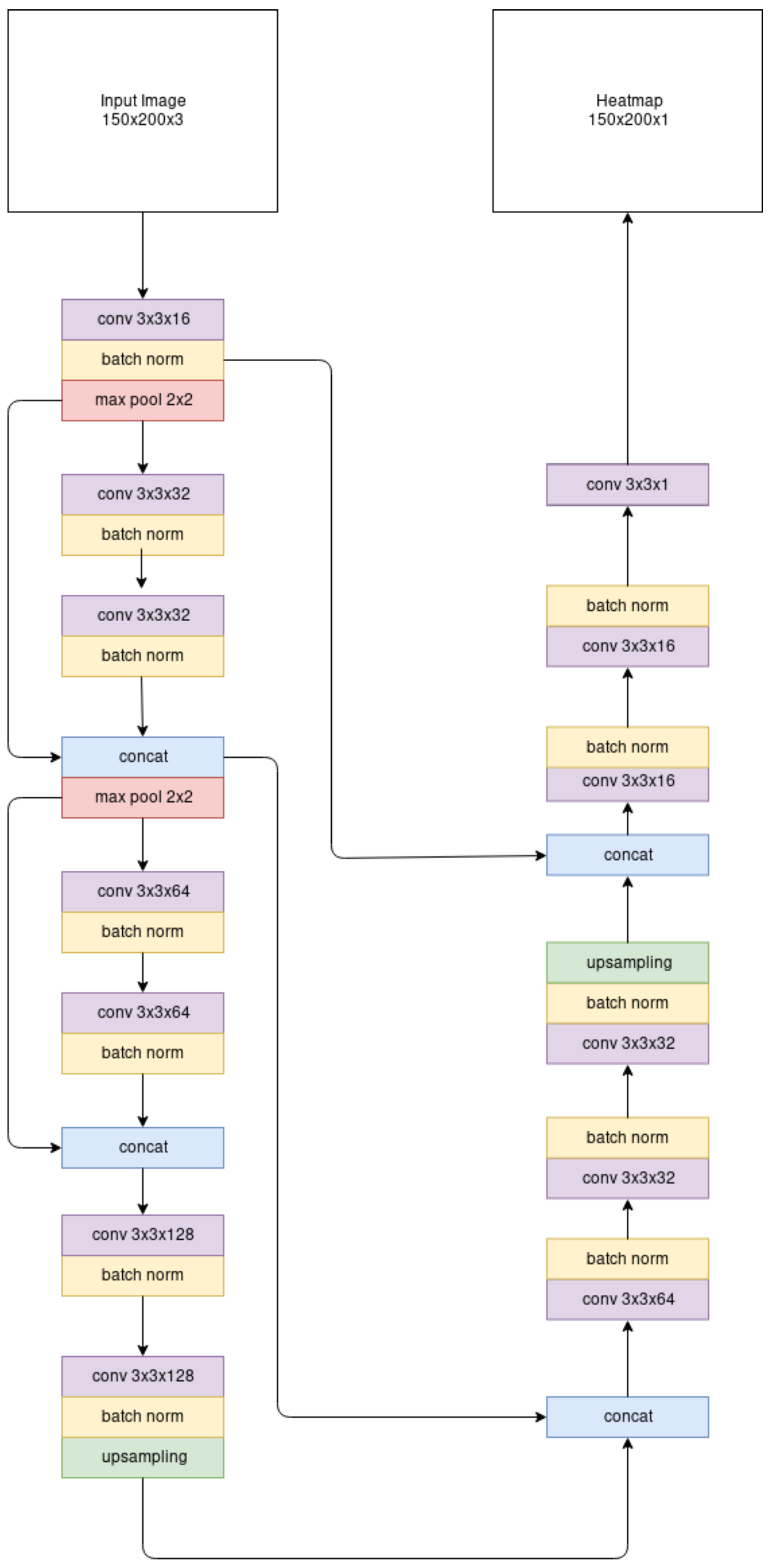

Figure 4.4.: The architecture for the neural network as proposed by Speck et al. [SBB18]. This architecture only generates one heatmap for the detection of one class.

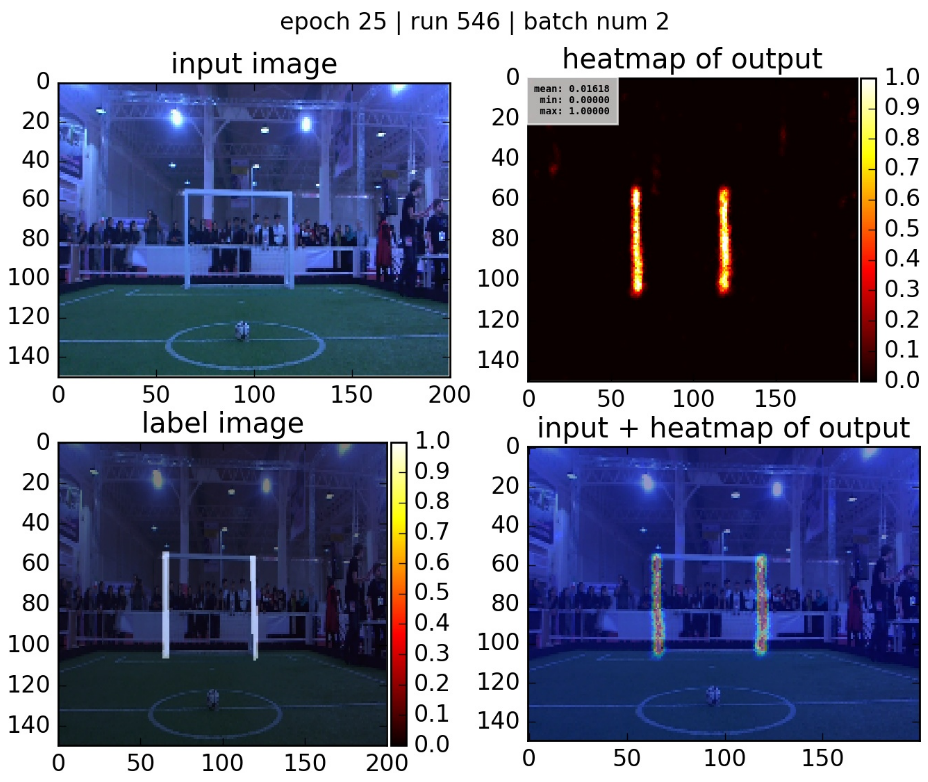

Figure 4.5.: Input image from [2]. This image shows two goalposts that have been correctly detected by the neural network. The goalposts have both been detected over the full length of the field and the detection is very accurate. There is no relevant activation in the output image other than the two goalposts.

4.7. One Channel - Bottom Part

Following the possible explanation given in sec. 4.6 a new approach was developed. The most relevant part of the goalpost for our case is the lowest point of the detected goalpost as that is the part that will be transformed from the image space to the relative space (see sec. 2.6 for details). Since detecting the full goalpost might lead to the problem of not properly detecting the bottom most part of the goalpost as seen in e.g. fig 4.6 this can lead to significant errors in the calculation of the position in relative space. If only the bottom part is detected it is unlikely to detect a part of the goalpost which is not on the ground, thus allowing for a more accurate calculation of the relative position.

Additionally, the work of van Dijk et al. [vDS18] showed their network not being able to reliably detect the full goalpost. Instead, they label a circle with a radius of 20 pixels at the position where the goalposts are on the field. This approach for labeling was

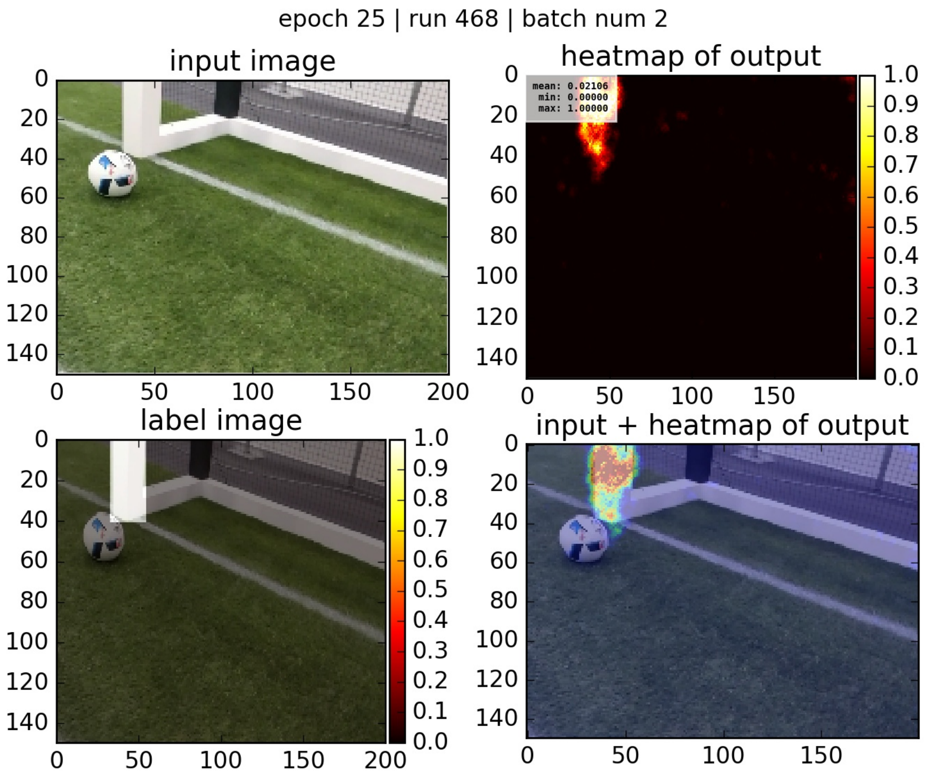

Figure 4.6.: Input image from [2]. A goalpost could also be detected from a closer distance which is a very important scenario in the RoboCup domain as the robot should be able to detect the goalpost even while being very close to the goal and in a position to score a goal. The goalpost is found even though it is right next to the ball which could lead to a false negative due to the training data mostly consisting of uncovered goalposts with no objects close to it. This is the case, because that is how the goalpost can be seen for most of the game.

not used in the proposed solution for this thesis. The labels were created as explained in 4.4 and then the top part of the label was cropped to only include the bottom part of the label. This solution was used so the labels already created could be reused and because this way the labels can be used for the approach of detecting the full goalpost as well as only the bottom part. For this method, the annotation values were first sorted by their y-values (the height). Then it was checked if e.g. the top part of the label of the left side was more than 10 pixels higher than the bottom part of the left side. If it passed this threshold of 10 pixels the height of this side was reduced by 90 percent of its height. Following this, the ground truth was created in the same way as explained in sec. 4.6 based on the smaller label.

The image 4.8 shows the detection of this approach working over a significant distance with robots blocking the view of one goalpost. In this example, an error of the way the label was cropped can be seen. There should either be no label for the right goalpost as the bottom part of the goalpost is completely occluded by the robot, or the right goalpost should be in the label completely and marked in the annotation that was humanly created with "concealed" as defined by the ImageTagger. This shows this approach of cropping the labels has downsides, however, for most of the images the labels were created correctly and thus this is not really a problem as is visible by the network also not having an activation at this position, while still detecting the left goalpost.

The fig. 4.9 shows difficulties with this approach. Since only the bottom part is learned the advertisement in the example at the top also has an activation of 0.5 which is not as high as the activation for actual goalposts shown in these example images, but still significant. This detection is likely due to the bottom part of the advertisement being a white spot on the image which is positioned right above the field. This might not be the case for the approach using the full goalpost as the ground truth since in that case the text of the advertisement would be different from what a regular goalpost looks like while this part is not relevant for the detection of the bottom part of the goalpost. Due to the advertisement in this example being right at the edge of the image there is also no more context to the side of the advertisement which might help the neural network not to detect this. The context is likely important, since the left part of the advertisement looks the same on the left and the right side, just being white, the left side has more context due to the image not being cut off and there is no activation on this side of the advertisement.

The bottom example in fig. 4.9 shows another problem with this approach. There are 2 true positives that should be detected in this image. The neural network has high activations at the position where the true positives should be. However, the back of the goal also has bars that look very similar to the goalpost, with the difference being they are at the back of the goal and not as high as a goalpost. For this part, a significant activation is visible at the bottom left in the image. On the other side of the goal where the same circumstances are true, there is no activation in the output. This might be due to the goalpost not being cut off, at the border of the image which allows for visible context. For example, the bar connecting the front of the goal with the back is completely visible where no activation is in the output, while most of the bar connecting the front to the back of the goal is not visible in the bottom part of the image space where the false positive detection occurred.

Another problem was shown in fig. 1.1 where the goalpost was detected, but also a part of the exhibition hall.

4.8. Two Channels

The approaches presented in sec. 4.6 and sec. 4.7 used the architecture presented in [SBB18], but instead of the ball learned the goalpost. It is still necessary to detect the ball however so this would require the layers to compute every layer twice since the

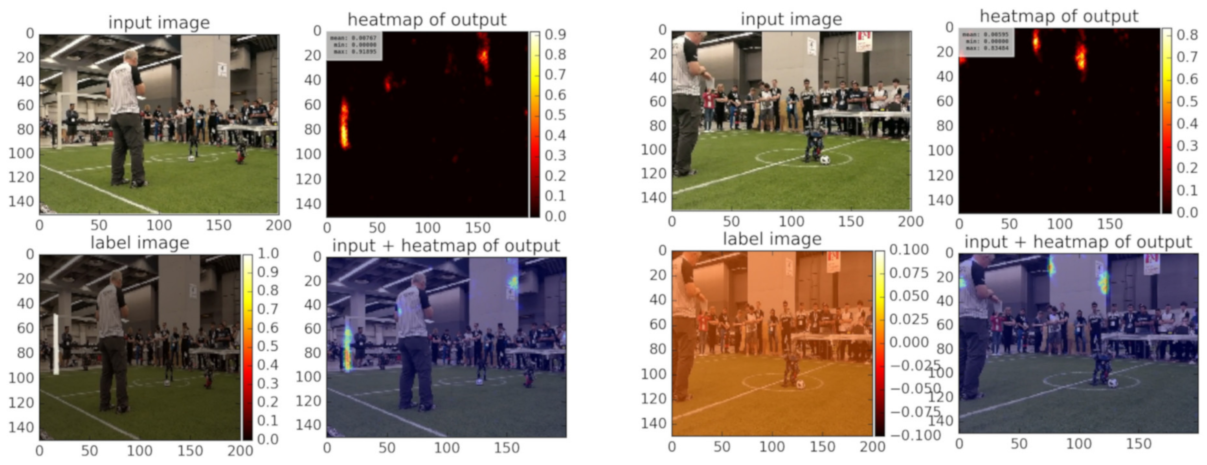

Figure 4.7.: Input images from [2]. Two images that show the neural network detecting false positives. In the left image, a pillar of the hall was also detected and in the middle of the detection of the pillar, the activation is significant. In the image on the right, no goalpost is present, but the output shows significant activations on the same pillar as well as a less significant detection on the shirt of the referee.

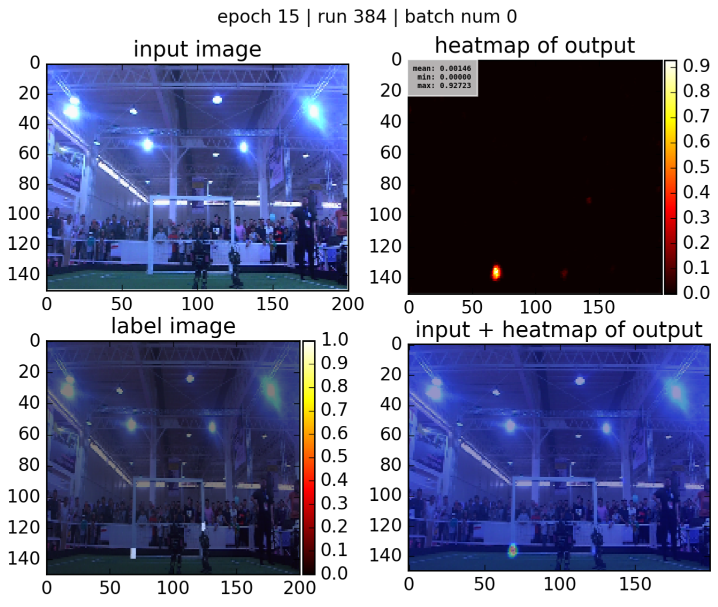

Figure 4.8.: Input image from [2]. Only the left goalpost is detected, which is the expected result, because the right goalpost is occluded by the robot and the bottom part is thus not visible.

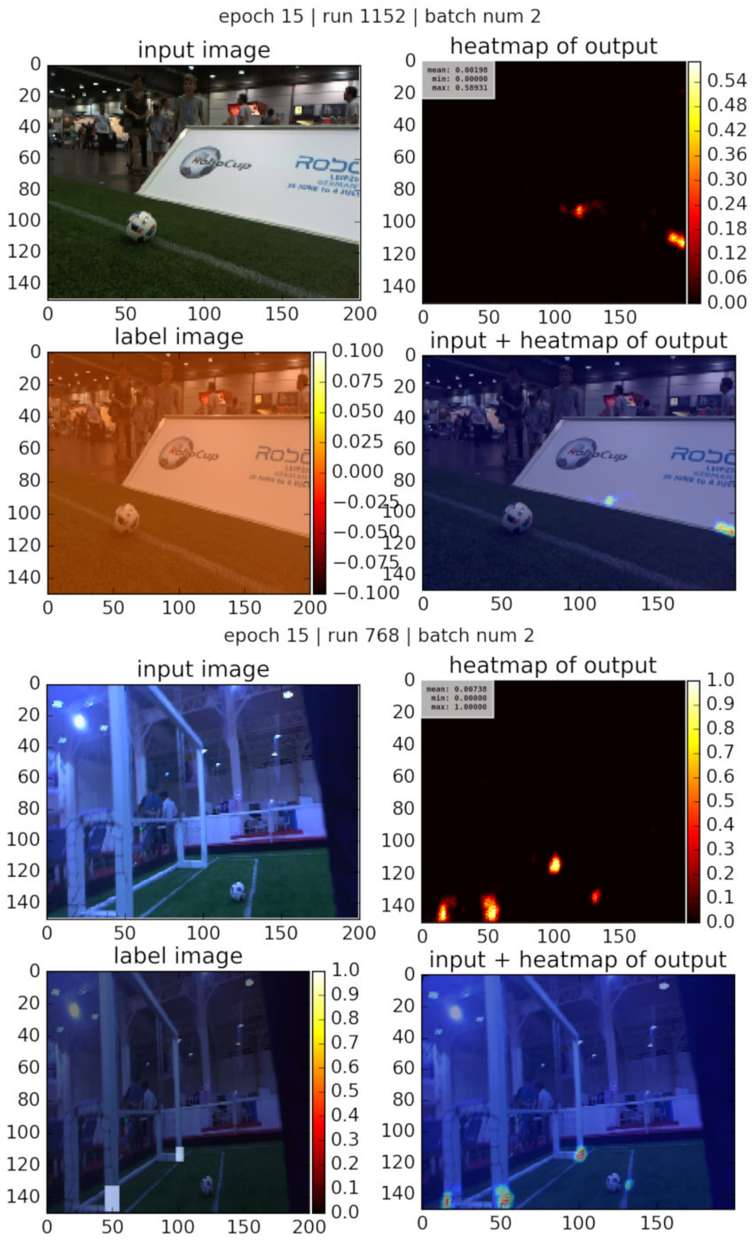

Figure 4.9.: Input images from [2]. In the **top** example, there are activations of more than 0.5 which is not very high, but still significant. In this example, the advertisement at the boundary of the field is detected, most likely because it is a white spot on the image which is placed on top of the green field. In the example at the **bottom**, 3 significant activations are visible. Two of them are on the actual goalposts while one is in the bottom left of the image space. This activation is on a part of the goal which looks like a goalpost with the difference being it is at the backside of the goal and it is shorter. This is hard to distinguish since especially the bottom part of the goalpost looks almost identical to the actual goalpost.

architectures would be completely distinct with these approaches. This is not feasible to use, because the computation of the neural network already takes up most of the time the vision pipeline takes to detect all of the objects in the image. If one of these approaches were used in a game with the robot the computation time for each frame would be significantly higher and the robots ability to react in real time to its environment would be significantly worse. One solution for this is to use the same architecture but add another output channel. This approach can be done by adding another filter at the last layer. This second filter means the output of the neural network is two heatmaps instead of one.

4.9. Two Channels - Bottom Part

The first approach of having two channels was done using only the bottom part of the layer as the ground truth. As explained in sec. 4.7 this approach could lead to better results of the calculation of the relative position of the goalpost to the robot. Another motivation were the results of van Dijk et al. [vDS18] which showed detecting the full goalpost was not reliably possible with their neural network since it had too many false positives of not distinguishing between background and goalposts enough.

The results in fig. 4.11 show this approach works well for the detection of the ball, with high activations at the position of the ball. However, the goalpost channel has almost no activation. The activation is less than 0.02 for all parts of the image even though two goalposts are visible in the upper input image.

Fig. 4.12 also shows the ball is detected, while the goalpost output channel has no activation present. The bottom of the figure shows input images from a different location and taken by a different camera. The ball is still detected well with an activation of up to 0.8, the goalpost present in the image is not recognized.

One reason for this could be the label of the goalposts being comparably small because they are cropped and also the goalpost is further away than the ball in most situations and training images which leads to smaller labels due to the goalpost being further away in relative space.

4.10. Two Channels - Full Goalpost

By using the full goalpost label as the ground truth instead of only using a small part at the bottom of the goalpost the problem in sec. 4.9 can be solved, because if the goalpost is not detected a larger ground truth of the goalpost leads to a larger error. Thus the weights are more fitted to detect the goalpost. This is intended to create a trade-off of detecting the ball not as accurately but tuning the filters of the neural network to also learn the features of the goalpost. This could be necessary because as pointed out in [vDS18] the goalposts are less feature-rich than the ball which makes it more difficult to detect the goalpost correctly.

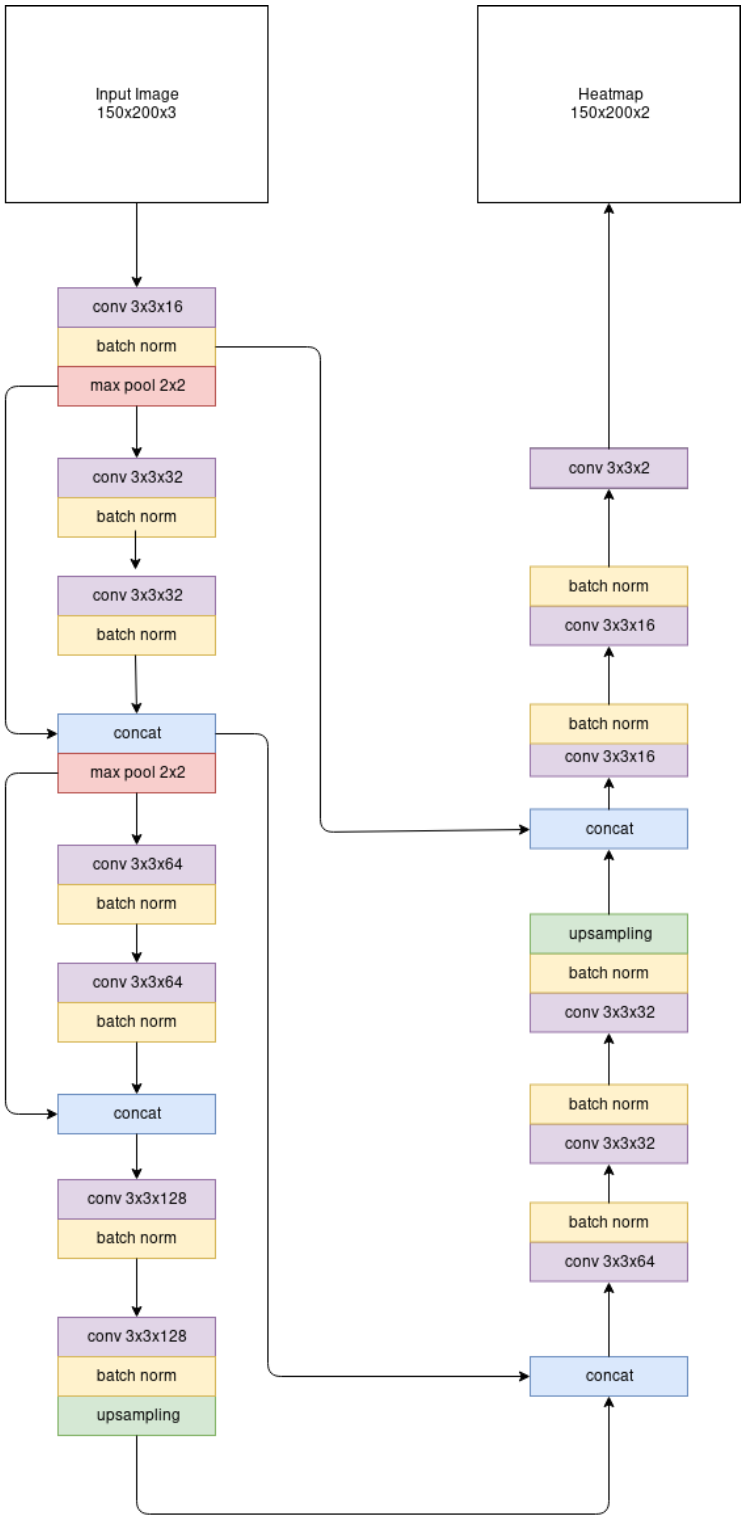

Figure 4.10.: The architecture used for the detection of the ball and goalpost. Another filter has been added in the last convolutional layer to produce another heatmap.

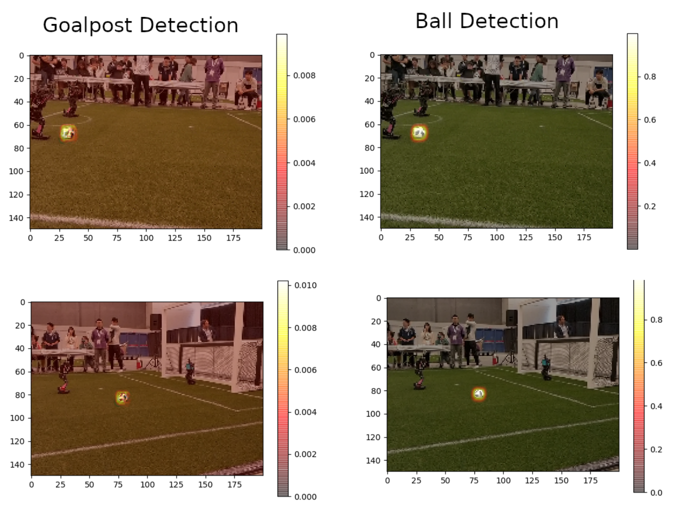

Figure 4.11.: Input image from [2]. The Images on the left are the output of the detection of the goalpost, while the images on the right are the output for the detection of the ball. The ball is detected with high confidence and no false positives are present. For the goalpost, there is almost no activation. There is only an activation of less than 0.02.

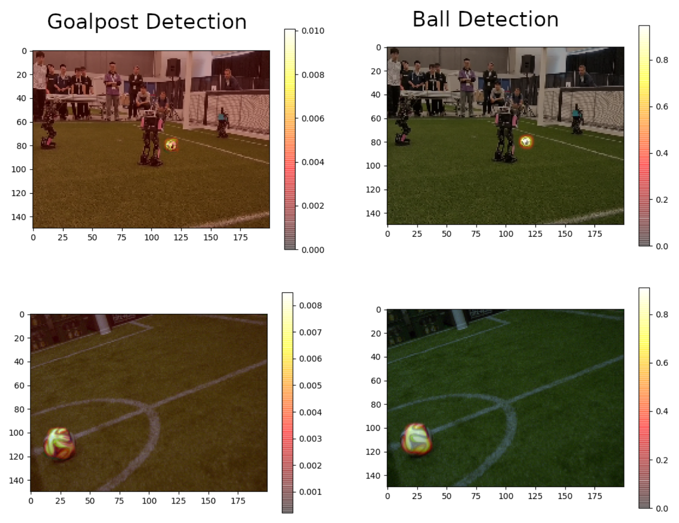

Figure 4.12.: Input image from [2]. As in fig. 4.11 the ball is detected with a high activation. The goalpost is not detected and there is no activation on the goalpost layer. The bottom two images are taken with a different camera and in a different location, but the goalpost is here also not detected, while the ball is detected.

Fig. 4.13 shows while false positives still happen there is an activation on both the goalpost and the ball channel. When a goalpost is present in the image it is detected accurately. The detection does not cover the full length of the goalpost, but the bottom part, which is the most important one for the transformation into relative space, is accurately detected. The false positive in this image could be filtered in post-processing by e.g. checking if the detected object is white and if the object meets the field boundary. Since the object in this image is not close to the field boundary, it could be ruled out as a goalpost.

Fig. 4.14 shows the neural network was able to adapt to a different camera with different settings for the pictures and still detect the goalpost which is not fully visible in the image as well as the ball.

Due to the ball being significantly more important in a game because it is the main object of play in soccer, another experiment was started which included more image sets which included mostly labels for the ball. This was done to change the results of the trade-off made to have a more accurate ball detection and a less accurate goalpost detection.

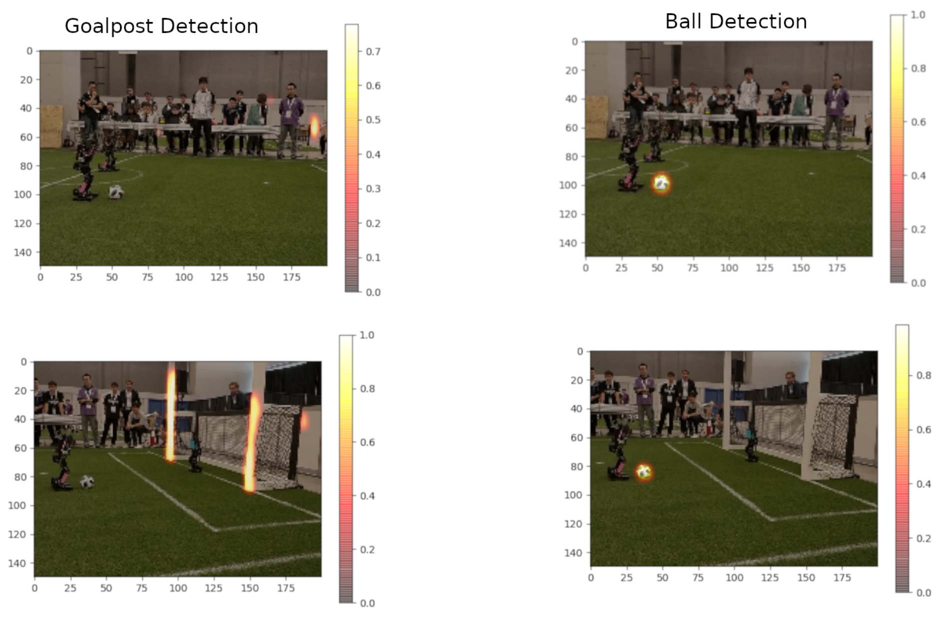

Figure 4.13.: Input images from [2]. The **top** image shows the goalpost detection on the left has activations. However, the activation is a false positive in this case with a significant value of up to 0.7. The ball on the right side is detected accurately with an activation of up to 1.0. On the **bottom**, the goalposts are detected. The detection does not cover the full height of the goalpost, but most of the goalpost is detected and the bottom part, which is the most important is detected. There is a small activation at the backside of the goal, it is not a high activation though and thus will be filtered out by a threshold requiring a minimum activation. The ball on the right is properly detected, again with high activations of up to 0.8

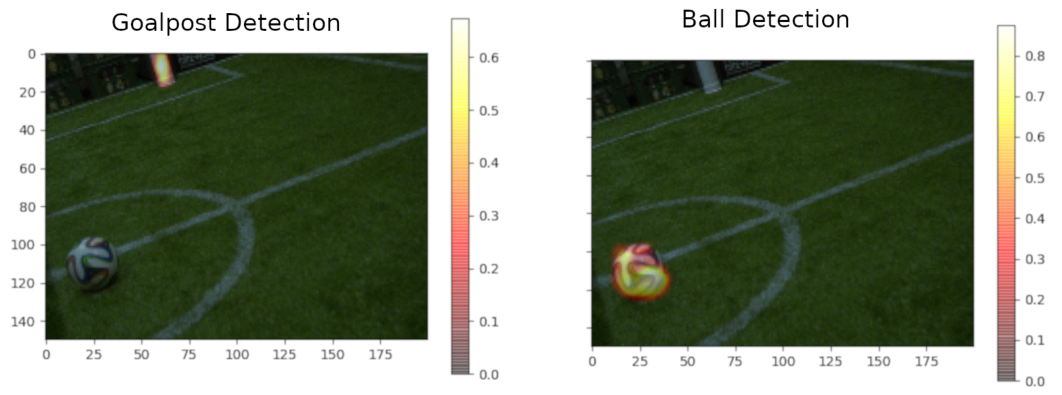

Figure 4.14.: The input image [2] is from a camera which was not used in the training set at all. Regardless the ball is detected with a high activation as well as the goalpost, of which only a small part is visible in the image.

5. Evaluation

This chapter evaluates the results of the presented approaches by calculating the intersection over union for the objects detected.
How to calculate intersection over union is explained in sec. 5.1. Additionally, there is a special case where the network has no activation passing the threshold and the object being not in the image. The section also explains why this is considered separately.
The image set used for the evaluation is presented in 5.2.
Data about the training sets including the used amount of labels for the training is described in 5.3.
The approach for using a single output channel and detecting the full goalpost is evaluated in 5.4, detecting the bottom part of the goalpost with a single output channel is evaluated in 5.5.
Detecting the ball as well as the goalpost in two channels is evaluated for the bottom part of the goalpost in 5.6, for the full goalpost in 5.7.
Using more image sets including 1299 ball labels with zero additional goalpost labels is evaluated in 5.8.
The results of all approaches are summarized in table C in the appendix.

5.1. Intersection over Union Calculation

The intersection over union (IOU) or Jaccard index is a metric used for the evaluation of the output of neural networks. It compares the manually labeled ground truth with the output of the neural network. The IOU is defined by the division of the intersecting area between the ground truth (L) and the output of the neural network (P) by the area covered by the ground truth label as well as the neural network output (see eq. 5.1).

$$\text{IOU} = \frac{|L \bigcap P|}{|L \bigcup P|} \tag{5.1}$$

For an FCNN this can be accomplished by applying a threshold on the output of the network. This means all activations in the output below e.g. 0.6 are regarded as no activation. Following this step, the values for each position in the image will be compared with a logical OR and a logical AND. The union is the number of positions where the logical OR has returned true, because in this position either the neural network output had an activation or the ground truth had a label in this position. The intersection is calculated by the number of positions where the logical AND has returned true since these are the positions where the ground truth and the neural network output overlap.

The intersection over union is calculated by dividing the value of the intersection by the value of the union. This means the highest value is 1 if the ground truth is the same as the prediction of the network. The worst value is 0 if there is no intersection between the ground truth and the prediction of the network.

The union can be of value zero if the object is not present in the image and the neural network had no activation. The value of the IOU for this case should be 1 since the prediction of the neural network is right. However this can skew the data, which is especially apparent in table 5.3, where the goalposts are not detected, but the mean IOU for the goalposts including the images with no goalposts is 28.6%. Due to this, the IOU for the objects is calculated including the cases where no object was present and the network predicted it correctly and without these cases. The statistic including these cases is called "including negative data" in the following tables.

5.2. Evaluation Image Set

The image set "bitbots-montreal-game02" [15] from the Hamburg Bit-Bots ImageTagger was chosen for evaluation, because it is the one used for the evaluation of the vision pipeline of the Hamburg Bit-Bots [FBG$^+$19] and thus the results are comparable for the ball and the goalpost detection. It was not used in the training of the network.

5.3. Training Sets

For the training of the networks, several image sets were selected from the ImageTagger of the Hamburg Bit-Bots [2]. The selected image sets are presented in table B.1 in the appendix in more detail. There were 25100 labels of goalposts present in the used training sets and 43058 labels of balls. For the training with more ball labels, 1299 additional ball labels were used while including no further goalpost label.

5.4. Single Channel - Full Goalpost

Table 5.1.: Single Channel - Full goalpost

IOU for Ball:	N/A
IOU for Goalpost:	46.3%
IOU including negative data Ball	N/A
IOU including negative data Goalpost	58.0%

The approach of using the full goalpost label with just one output channel had an IOU for the goalpost of 46.3% and if the negative data for the goalpost is included an IOU of 58%, as shown in table 5.1. The results are significantly worse than the results for the ball detection with the same architecture [SBB18]. This is likely due to the goalpost being less feature-rich than the ball. The mean IOU increasing by 11.7% by including the images where no goalpost was present in the image and the network had no detection, shows the network detects true negatives in the image set.

5.5. Single Channel - Bottom Part of Goalpost

Table 5.2.: Single Channel - Bottom Part of Goalpost

IOU for Ball:	N/A
IOU for Goalpost:	27.1%
IOU including negative data Ball	N/A
IOU including negative data Goalpost	48.0%

The IOU for detecting the bottom part of the goalpost, as shown in table 5.2 is significantly worse than the IOU for detection of the full goalpost. A part of this is likely due to it being hard to detect how much of the goalpost is the bottom part as defined by the ground truth. If the detection is a little bit smaller than the ground truth, the goalpost would be detected correctly, but the IOU would still be smaller. Additionally, the example in fig. 4.8 shows the manual labeling process to create the ground truth was not always performed correctly, which will further lead to worse results of the IOU. The results show a 20.9% increase in the IOU if negative data is included in the IOU calculation, which shows though fewer true positives were detected, true negatives were often detected correctly.

The low value of the IOU for this experiment shows it was harder to detect the bottom part of the goalpost for the architecture with the image sets that were used for training.

5.6. Two Channels - Bottom Part of Goalpost

Table 5.3.: Two Channels - Bottom Part of Goalpost

IOU for Ball:	73.9%
IOU for Goalpost:	0.0%
IOU including negative data Ball	73.9%
IOU including negative data Goalpost	28.6%

The results of using two channels as output while learning the bottom part of the goalpost are shown in table 5.3. The IOU for the goalpost is 0.0% because as was shown in sec. 4.9, the goalpost channel had no activations and thus there was no intersection between the ground truth and the output of the neural network for the goalpost. This is likely, because as shown in table 5.2 compared to table 5.1 the bottom part of the goal-post is significantly harder to learn and thus the network only detects the easier ball and ignores the detection of the goalpost. The IOU for the goalpost including negative data was, however, 28.6%, because every true negative in the image set was correctly detected as such, as there was also no activation for these input images. The ball is detected with an IOU of 73.9%, which is better than the detection of the ball in the vision pipeline of the Hamburg Bit-Bots [FBG+19], which is likely due to including other image sets than the ones used in [SBB18].

5.7. Two Channels - Full Goalpost

Table 5.4.: Two Channels - Full Goalpost

IOU for Ball:	65.8 %
IOU for Goalpost:	40.3%
IOU including negative data Ball	65.8%
IOU including negative data Goalpost	49.3%

The detection of the goalpost with 40.3% is significantly better than the results of the current Hamburg Bit-Bots vision pipeline of 18.3% [FBG+19], with the IOU of the ball being slightly worse than 67.7% [FBG+19]. The small reduction of the detection of the ball is because the task of detecting the ball and the goalpost is harder than just detecting the ball. However the results are not significantly worse, this is likely due to being able to use feature maps of the convolutional layers for the goalposts as well as for the ball.

5.8. Two Channels - Full Goalpost with more Ball Labels

Table 5.5.: Two Channels - Full Goalpost with more Ball Labels

IOU for Ball:	69.7%
IOU for Goalpost:	24.9%
IOU including negative data Ball	69.7%
IOU including negative data Goalpost	40%

Including more image sets featuring mainly labels of the ball has led to a significant decrease in IOU for the goalpost. However, the ball is now detected more accurately than without these labels and more accurately than the results of the current vision pipeline of the Hamburg Bit-Bots [FBG+19] with 67.7%.

5.9. Runtime Difference

The runtime for the two architectures was measured by computing the prediction of the network in the vision pipeline of the Hamburg Bit-Bots and measuring the time before the function call and after the function call. To not have outliers influence the values computed this was measured for more than 200 times for both architectures. The first runtime value was removed, due to the startup taking multiple seconds and not being relevant because the robot will be finished with the startup by the time it is put in the game. The resulting values are shown in table 5.6.

Table 5.6.: Runtime Difference for Architecture with One Channel and Two Channels

Average One Channel Runtime	0.0494 seconds
Average Two Channels Runtime	0.0497 seconds
Median One Channel Runtime	0.0463 seconds
Median Two Channels Runtime	0.0483 seconds

The difference in the average runtime for one and two output channels is 0.0003 seconds, or 0.3 milliseconds. This is an increase in runtime of 0.7%.

The difference of the median is 0.002 seconds or 2 milliseconds.

The difference in runtime is computing the last feature map that was added as well as transferring the additional output channel from the GPU of the Jetson TX2 to the CPU. The difference in runtime is insignificant.

6. Discussion

This chapter discusses the decisions made in the process of the bachelor thesis.

The sec. 6.1 will discuss which image sets were used in the training process as well as which problems were noticed with the available image sets. Sec. 6.2 discusses the choice of image set for the evaluation. Deciding which part of the goalpost or the full goalpost should be the ground truth is discussed in sec. 6.3, while considering the results of Van Dijk et al. [vDS18]. Whether images containing only ball or only goalposts labels should be included in the training set is discussed in sec. 6.4.

6.1. Training Data Selection

The training images mostly consist of images that were taken by a human with a phone at the border of the playing field. This is not optimal since the perspective of the robot will be different. The robot is smaller than the human which means the robot will have a lower angle to view his environment. Additionally, the robot will look to the ground in front of him most of the time, because he will be searching for a ball. This means the robot will comparably less often be able to see the full goalpost, it will most of the time see either no goalpost at all or only the bottom part of the goalpost.

The training data was chosen this way because only few image sets were taken from the perspective of the robot and the labels for the goalpost and ball were mostly available for image sets taken from the side of the playing field.

More images taken from the robot should have been created and labeled to ensure the applicability of the neural network to the RoboCup Soccer domain.

The image sets used in the training for the network evaluated in 5.8 only contained labels of balls. This meant the balls were detected better, however it also led to a significantly worse detection of the goalposts. This could likely be avoided by having labels for the goalpost also included in the image set.

6.2. Evaluation Image Set

The image set bitbots-montreal-game02 [15] in the Hamburg Bit-Bots ImageTagger [2] was chosen as the evaluation image set. This decision was made so the results are directly comparable to the results published in [FBG+19], which are the results of the current vision pipeline of the Hamburg Bit-Bots. This image set provides a variety of angles of the ball and goalposts with moving robots. The images are also from a real game that was played in the world championship in Montreal in 2018. With a pillar visible in the background as well as another field, the detection task is as hard as with a normal game in the RoboCup Soccer domain unlike a lab environment would be.

However the images were all taken from the side of the field, so a larger part of the goalposts is visible than the images of the robot would likely be able to capture. Additionally, it only contains images from a single game which means overfitting for the exhibition halls or the lighting in the Montreal exhibition hall would not be reflected in the results of the evaluation.

6.3. Full Goalpost or Bottom Part of Goalpost

Van Dijk et al. [vDS18] had difficulties detecting the full goalpost because for the full goalpost the network was not able to discern between the goalpost above the field boundary and the background of the goalpost. The results presented in this thesis had difficulties detecting only the bottom part of the goalpost while detecting the full goalpost worked significantly better.

Van Dijk et al. have only used one image set for the training of the ball detection. If the same image set was used for the training of the goalposts, then the variation in backgrounds is limited, because only images from one field in one location were used. This could explain why the network had difficulties generalizing from this data.

The used data for this thesis contained significantly more ball labels than goalpost labels as mentioned in sec. 5.3. The bottom of the goalpost is less feature-rich than the full goalpost since it is only a white spot instead of a longer vertical white bar in the image. Thus the network in this thesis was not able to solve the problem of detecting the bottom of the goalpost. For the network to be able to detect the bottom of the goalpost, the perspective the robot will have most of the time, more training data would likely help to learn this less feature-rich part.

6.4. Filter Images Based on Available Labels

Not for every image in the training set labels for the ball as well as the goalpost were available. One option of handling this was to only include images for the training where a label was available for a ball as well as a goalpost. Either a label encompassing the object or a label marking the object as not visible in the image. This leads of course to less available training images. This approach was tested early on in the training for two channels and showed the network was significantly worse in detecting objects because less diverse training data was available.

Thus the decision was made to include all images containing either a ball or goalpost label. This meant the ground truth image was a black heatmap where no object was labeled no matter if the object is present in the image or not. While this will lead to the weights being changed to avoid false positives, even though the detection was correct, there is significantly more training data available, especially for the ball.

7. Conclusion and Future Work

In sec. 7.1 the thesis is concluded. The best approaches are presented and compared to the current state of the Hamburg Bit-Bots vision pipeline. Possible future work is mentioned in sec. 7.2.

7.1. Conclusion

The evaluation in sec. 5.7 showed the approach of using two channels in the network has an IOU of 65.8 for the ball detection and 40.3% for the detection of the goalpost. The current vision pipeline of the Hamburg Bit-Bots has an IOU of 67.7% [FBG$^+$19] for the detection of the ball on the same image set as the one used in the evaluation of this thesis. This means the detection of the ball is slightly worse, however, the detection of the goalpost in the current vision pipeline has an IOU of 18.3% [FBG$^+$19]. This shows a significant increase in the goalpost detection with only a small difference in the ball detection. With the runtime of the prediction of the neural network on average only taking 0.3 milliseconds longer, this difference is negligible. With post-processing, the results could be improved even more by filtering out obvious false positives such as balls over the field boundary. Thus in the future, the neural network could be used in the vision pipeline of the Hamburg Bit-Bots.

The approach by Van Dijk et al. [vDS18] used a different image set for the evaluation than the one used in this thesis and is only detecting the bottom of the goalpost. Thus the results are not directly comparable. The paper had two networks where additionally training goalposts was also evaluated. The IOU for the network with the higher IOU was 75.4% for the ball, with the goalpost being detected with 27.3%. The network in sec. 5.7 had a worse performance for the ball, while the goalposts were significantly better detected.

Detecting the goalpost with a single output channel had an IOU of 46.3%, which is better than the approach with two channels, however this is only an increase of 6% and the increased runtime of running the network once for the ball and again for the goalpost is not feasible due to the real-time constraint of the robots.

The results presented in this bachelor thesis show the architecture proposed by Speck et al. [SBB18] is capable of detecting additional object classes other than the ball by adding another output channel. The cost in runtime of adding another output channel and the trade-off in accuracy are not significant.

7.2. Future Work

This section discusses future work which could build on the results of this thesis and improve them further.

The loss function that is currently used could be improved in the future. The current loss function calculates the loss by calculating the squared difference between the prediction of the network and the ground truth. The size of the label could be included to make a small object become more significant to make sure objects further away are also detected. This could, however, lead to objects which are close having a big bounding box to being less well detected.

The loss of detecting the goalposts or the ball could be multiplied to value the detection of one object over another. This could be used to detect less feature-rich objects by valuing them more in the learning process.

The IMU measurements available in the robot could be used to generate an image that is not tilted unlike the image taken by the robot. This would lead to goalposts always being a straight vertical line which might be easier to detect. However, this would also mean the result of the IMU sensor would be required in the vision pipeline which would add another dependency to the vision pipeline and time would be required to wait for the IMU sensor data and to generate an image that is not tilted.

If the input image to the neural network is not tilted a bounding box approach such as YOLO [RF17] could be used. While the result will not be pixel precise, multiple teams in the RoboCup Soccer domain use this approach and are able to also detect e.g. robots with this method.

As discussed in sec. 6.1, the training data consists mostly of data that was recorded by a human at the side of the field. However, training data captured by a robot would be more realistic for the domain, since these images would be more similar to images to other images the robot will see. Thus more images should be recorded this way and labeled to then be used as training sets.

The labels are currently generated by humans by selecting four points per goalposts. This is a task that requires a lot of time if a lot of labels are required. One solution for this would be to take images with AprilTags [Ols11, WO16] in a known position relative to the robot. AprilTags are a system where targets can be printed and then their position, orientation and identity are easily detectable with a camera. AprilTags are significantly easier to detect in the image than goalposts. By knowing the position of the AprilTags, the goalposts could be automatically labeled, because the goalposts do not move and thus the relative position of the goalposts to the AprilTags would be enough to generate the label. However, with this approach, the neural network would likely learn to find the feature-richer AprilTag rather than the goalpost.

One approach that would solve the problem of detecting the AprilTags instead of the goalposts would be using two cameras looking into different directions. The cameras would be calibrated in position to each other by e.g. having the same AprilTag in one image for both cameras and then calibrating the relative position of the cameras to each other by calculating where the AprilTag is relative to the cameras. This way the AprilTag would not be in the image with the goalposts, but the relative position of the AprilTag to the goalposts would still be known. With this method, the goalposts could be automatically labeled while having no AprilTag in the image. This would only require moving the camera to generate thousands of images with known positions of the goalposts.

Another method to generate labels would be to use a different neural network which is not required to run in real time on a robot to generate labels. This network could take longer for the detection of the objects but be more precise. The predictions could be clustered and uploaded to the ImageTagger [FBH18] which provides a verification function so humans could verify if the label is accurate or not. This would provide a significantly faster way to generate data than manually creating the labels.

The trained network could be optimized for the use on a robot with e.g. TensorFlow Lite [16] which is likely to make the prediction faster. This means a deeper network could be used, since the computation time with optimizations would likely not take much longer. This would allow for a more accurate detection while not taking longer for the computation.

A visual compass is currently being developed by the Hamburg Bit-Bots team. The idea is to find features in the image which will not change during a game outside of the field boundary for detecting the orientation of the robot on the field. The output of the visual compass could be taken into consideration to filter false positives of the neural network because the robot can not see a goalpost on his field if it is looking to the side of the field.

Since this thesis has shown the neural network is able to detect more than a single class it could be evaluated if the architecture is also able to find more object classes like field markings. This would likely not significantly influence the results of the goalpost and ball detections but could help with the self-localization.

Acknowledgments

Thanks to the RoboCup team Hamburg Bit-Bots, and especially to Marc Bestmann and Dr. Norman Hendrich. I am also thankful to the NVIDIA Corporation for supporting the research of the Hamburg Bit-Bots via the NVIDIA GPU Grant Program. The donated NVIDIA Titan X (Pascal) was used to train the models trained for this thesis.

Appendices

A. Export Formats

A.1. Imagetagger Export Format for Goalposts

```
Name: goalpostToYaml
Name format: export_%%exportid.txt
Public: False
Annotations types: AnnotationType: goalpost
Minimum Amount of Verifications needed: 0
Include blurred annotations: True
Include concealed annotations: True
Base format: set: %%imageset
            images:
            %%content
Group annotations by images: True
Image format:     %%imagename:
                        width: %%imagewidth
                        height: %%imageheight
                        annoamount: %%annoamount
                        annotations:
                %%annotations
Annotation format:
                    -
                  %%vector
Vector format:               - [%%x,%%y]%%br
Not in image format:         -
                                                   - notinimage
```

A.2. Imagetagger Export Format for Goalposts and Balls

```
Name: goalpostToYaml
Name format: export_%%exportid.txt
Public: False
Annotations types: AnnotationType: goalpost
                   AnnotationType: ball
Minimum Amount of Verifications needed: 0
Include blurred annotations: True
Include concealed annotations: True
Base format: set: %%imageset
images:
    %%content
Group annotations by images: True
```

```
Image format:      %%imagename:
                   width: %%imagewidth
                   height: %%imageheight
                   annoamount: %%ann%%annotations
Annotation format:
                     -
                     %%vector
Vector format:                - [%%x,%%y]%%br
Not in image format:         -
- notinimage
```

B. Training Sets

Table B.1.: The image sets used in the training of the neural networks including the number of goalpost and ball labels present in these image sets. The image set ID is the id which is used in the Hamburg Bit-Bots ImageTagger. It can be accessed by visiting the following url: `https://imagetagger.bit-bots.de/images/imageset/{id}/` where {id} is replaced by the id in the table.

Image Set ID	Number of Goalpost Labels	Number of Ball Labels
6	1,508	1,009
25	151	1,013
29	125	927
32	118	953
176	401	3
179	279	14
180	945	3
347	1,307	0
374	2,766	0
5	182	1,002
7	117	1,009
13	140	1,000
14	127	1,000
15	156	1,000
16	142	1,012
18	137	1,004
33	115	1,006
36	647	5,612
81	887	1,832
160	280	207
161	280	0
162	859	976
166	1,859	2,045
184	952	747
186	940	876
189	2,825	1,750
35	1,592	2,314
168	198	333
260	730	6,948
12	115	1,003
17	118	1,034
30	105	936
31	103	896
185	628	579
188	224	219
191	2,635	2,769
353	407	0

Table B.2.: The additional training sets that were included in the training of the network with more balls.

Image Set ID	Number of Goalpost Labels	Number of Ball Labels
153	0	283
156	0	1,016

C. Evaluation of All Approaches

	Single Channel Full Goalpost	Single Channel Bottom Part	Two Channels Bottom Part	Two Channels Full Goalpost	Two Channels Full Goalpost More Ball Labels
IOU for Ball	N/A	N/A	73.9%	65.8%	69.7%
IOU for Goalpost	46.3%	27.1%	0.0%	40.3%	24.9%
IOU including negative Data for Ball	N/A	N/A	73.9%	65.8%	69.7%
IOU including negative Data for Goalpost	58.0%	48.0%	28.6%	49.3%	40%

Bibliography

[AGH+19] J. Allali, L. Gondry, L. Hofer, P. Laborde-Zubieta, S. N'Guyen O. Ly, G. Passault, and Q. Rouxel A. Pirrone. Rhoban football club - team description paper humanoid kid-size league, robocup 2019 sydney. *RoboCup Symposium*, 2019.

[BBE+19] Marc Bestmann, Hendrik Brandt, Timon Engelke, Niklas Fiedler, Alexander Gabel, Jasper Güldenstein, Jonas Hagge, Judith Hartfill, Tom Lorenz, Tanja Heuer, et al. Hamburg Bit-Bots and WF Wolves team description for robocup 2019–humanoid kidsize. *RoboCup Symposium*, 2019.

[BCBK+19] Dan Barry, Andrew Curtis-Black, Merel Keijsers, Munir Shah, Matthew Young, Humayun Khan, and Banon Hopman. Electric Sheep team description paper humanoid league kid-size 2019. *RoboCup Symposium*, 2019.

[FBG+19] Niklas Fiedler, Hendrik Brandt, Jan Gutsche, Florian Vahl, Jonas Hagge, and Marc Bestmann. An open source vision pipeline approach for robocup humanoid soccer. In *RoboCup 2019: Robot World Cup XXIII*. Springer, 2019. accepted.

[FBH18] Niklas Fiedler, Marc Bestmann, and Norman Hendrich. Imagetagger: An open source online platform for collaborative image labeling. In *RoboCup 2018: Robot World Cup XXII*. Springer, 2018.

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GHSG18] Alexander Gabel, Tanja Heuer, Ina Schierin, and Reinhard Gerndt. Jetson, where is the ball? using neural networks for ball detection at robocup 2017. *RoboCup 2018: Robot World Cup XXII*, 2018.

[Gü19] Jasper Güldenstein. Comparison of measurement systems for kinematic calibration of a humanoid robot. Bachelor Thesis at the University of Hamburg, 2019.

[HHM$^+$19] Riki Hayashi, Yasuo Hayashibara, Hideaki Minakata, Kiyoshi Irie, Joshua Supratman, Youta Seki, Chisato Kasebayashi, Satoshi Shimada, Takaharu Nakajima, Shuto Takano, Naoki Takahashi, Kanta Takasu, Koki Matsumoto, Morito Ito, Takumi Ogasawara, Nguyen Anh Quan, Shuta Takahashi, Takehiro Hasegawa, Satoru Negishi, Gaku Kuwano, Shunsuke Takami, Hayato Kitaura, Ryoko Shiojima, Keito Yamada, Ema Tamamizu, Kazuku Nakajima, Yuki Kimura, Shigechika Miki, Yoshitaka Nishizaki, Kenji Kanemasu, Hajime Sakamoto, and Shuichi Yamaguchi. Cit brains (kid size league). *RoboCup Symposium*, 2019.

[Kan11] Mehmed Kantardzic. *Data Mining: Concepts, Models, Methods, and Algorithms*. John Wiley & Sons, second edition edition, 2011.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[LVMR19] José Pedro Villafan Luna, Salvador Guadalupe Rojas Vázquez, Ing. Julio Cesar Juárez Martinez, and Ing. Daniel Neri Ramírez. Eaglebotx.mx team description paper robocup 2019 sydney, australia. *RoboCup Symposium*, 2019.

[Ols11] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407. IEEE, 2011.

[RAS$^+$17] Eko Rudiawan, Riska Analia, P Daniel Sutopo, Hendawan Soebakti, et al. The deep learning development for real-time ball and goal detection of barelang-fc. In *2017 International Electronics Symposium on Engineering Technology and Applications (IES-ETA)*, pages 146–151. IEEE, 2017.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[RF17] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[SBB18] Daniel Speck, Marc Bestmann, and Pablo Barros. Towards real-time ball localization using cnns. *Robot World Cup XXII. Springer*, 2018.

[SHK$^+$14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[SSW$^+$17] Fabian Schnekenburger, Manuel Scharffenberg, Michael Wülker, Ulrich Hochberg, and Klaus Dorer. Detection and localization of features on a soccer field with feedforward fully convolutional neural networks (fcnn) for the adult-size humanoid robot sweaty. In *Proceedings of the 12th Workshop on Humanoid Soccer Robots, IEEE-RAS International Conference on Humanoid Robots, Birmingham*, 2017.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[vDS18] Sander G van Dijk and Marcus M Scheunemann. Deep learning for semantic segmentation on minimal hardware. *RoboCup 2018: Robot World Cup XXII*, 2018. to appear.

[WO16] John Wang and Edwin Olson. Apriltag 2: Efficient and robust fiducial detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198. IEEE, 2016.

Online References

[1] "Robocup soccer humanoid league laws of the game (draft!)." http://www.robocuphumanoid.org/wp-content/uploads/RCHL-2018-Rules-Proposal.pdf. last accessed: 2019-05-23.

[2] "Bit-bots imagetagger." https://imagetagger.bit-bots.de/images/. last accessed: 2019-06-14.

[3] "The official website of the robocup humanoid league." https://humanoid.robocup.org/. last accessed: 2019-06-13.

[4] "Robocupsoccer - small size." https://www.robocup.org/leagues/7. last accessed: 2019-06-13.

[5] "Robocupsoccer - middle size." https://www.robocup.org/leagues/6. last accessed: 2019-06-13.

[6] "Robocupsoccer - standard platform." https://www.robocup.org/leagues/5. last accessed: 2019-06-13.

[7] "Robocupsoccer - humanoid." https://www.robocup.org/leagues/3. last accessed: 2019-06-16.

[8] "A basic introduction to neural networks." http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html. last accessed: 2019-06-09.

[9] K. Hinkelmann, "Neuralnetworks." http://didattica.cs.unicam.it/lib/exe/fetch.php?media=didattica:magistrale:kebi:ay_1718:ke-11_neural_networks.pdf. last accessed: 2019-06-09.

[10] "Robots of the Hamburg Bit-Bots and WF Wolves." https://submission.robocuphumanoid.org/uploads//Hamburg_Bit_Bots_and_WF_Wolves-specs-5c03d58ec8f93.pdf. last accessed: 2019-06-14.

[11] Nvidia, "Jetson tx2 module." https://developer.nvidia.com/embedded/buy/jetson-tx2. last accessed: 2019-06-14.

[12] "Yolo: Real-time object detection." https://pjreddie.com/darknet/yolo/. last accessed: 2019-06-09.

[13] "The official yaml web site." https://yaml.org/. last accessed: 2019-06-13.

[14] "Welcome to pyyaml." `https://pyyaml.org/`. last accessed: 2019-06-13.

[15] Bit-Bots, "bitbots-montreal-game02." `https://imagetagger.bit-bots.de/images/imageset/261/`. last accessed: 2019-06-15.

[16] TensorFlow, "Tensorflow lite." `https://www.tensorflow.org/lite/`. last accessed: 2019-06-22.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Bachelorstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel — insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen — benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, den 25.06.2019

Jonas Hagge

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den 25.06.2019

Jonas Hagge