Weakly Supervised Object Detection in RoboCup Scenarios

bachelor thesis of

Maximilian Birkenhagen
matriculation-No: 6948018
bachelor: informatics

Department of Informatics, University of Hamburg
First supervisor: Prof. Dr. Simone Frintrop
Second supervisor: Ge Gao

Abstract

In this thesis, weakly supervised learning methods are tested in the RoboCup context. To make pixel-precise predictions using convolutional neural networks, pixel-precise data sets are usually required for training. However, by using recursive training and an approach that uses the GrabCut algorithm, pixel-precise predictions can be made from bounding box labels without changing the neural network structure. While the GrabCut method did not achieve good results, the last iteration of the recursive training approach had a mean IoU of 73,32% for the target class ball. Compared to the result of the fully supervised pixel-precise learning method with a mean IoU of 75,42% for the target class ball, this weakly supervised method was only about 2% worse and therefore competitive. Neither the fully supervised approach nor the weakly supervised methods could achieve satisfactory results for detecting goalposts and robots.

Keywords— object detection, weakly supervised learning, RoboCup

Contents

1 Introduction

This chapter introduces the work by looking at the motivation and possible benefits behind the use of weakly supervised learning. Furthermore, the goal of the research is explained. In this thesis, weakly supervised learning is applied by modifying the labels used during the training of neural networks many different ways, which are explained in detail in chapter 4.

1.1 Motivation

In the computer vision field, the main goal is to extract information from images and video sources [Hua96]. The knowledge we try to gather differs depending on the application. For example, if we want to take a portrait of a person using a camera, one such task could be to automatically focus on the face of the person to be photographed. Another use would be to provide the necessary information to enable self-driving vehicles [Boj+16]. The way to achieve this relies on object detection, an approach that describes the task to localise each instance of certain object classes in an image [Hal+20]. Because of the good performance of deep learning approaches on various tasks, deep learning methods are suited much better than the traditional methods. Therefore it is no surprise that the common state of the art for object detection relies on using deep neural networks and on applying several supervised learning practices [Pap+15].

While effective, one big drawback of supervised learning is the amount of data required [Kho+17]. Large amounts of annotated data sets are needed to successfully train such a neural network. This annotation process needs to be done manually and depending on the field can only be done by human experts of the aforementioned field [Kho+17; BBU18; Zho17]. Additionally, manual labelling also requires a lot of time. While a low-level image classification label, which only assigns a class id to the whole image, takes a mere second, high-level pixel-precise annotations, which assign class ids to each pixel in an image, can take up to 78 seconds per object instance [BBU18].

To annotate a whole data set with those high-level features takes way to long for every field of use. Also, for some areas, not even human experts can accurately depict the correct labels [Shi+16]. That means we may don't even have the needed data to train a neural network in a supervised manner in the first place. For example, in the medical domain, it is often difficult to identify the exact location of tumour cells in x-ray images and MRT scans, which is necessary for successful surgery operations [Shi+16].

In contrast, sparse or low-level labelled data is easy to obtain [Zho17]. Either one uses one of several already existing data sets and takes advantage of many sources like flickr.com and kaggle.com, or one can create own data sets that follow the cost structure outlined above. As a result, problem fields where only coarse classifications are needed, are already studied plenty and with significant success [Che+18]. Therefore, the next step in is to get the same good results for more detailed object detection.

The RoboCup, short for Robot World Cup, is an annually held competition, where mobile robots are The RoboCup (Robot World Cup) is an annual competition in which mobile robots compete against each other in soccer matches. It was proposed by Hiroaki Kitano et al. [Kit+95] as a standard problem to research new technologies for robots and artificial intelligence. The Hamburg Bit-Bots [Bitb] are a team of students from the University of Hamburg, who build soccer-playing robots within the scope of the RoboCup Federation [Fed19]. In the team, the students not only assemble the robots but also tweak their software to advance not only the results of their robots but also gain insight from their respective scientific fields. The ultimate goal of the RoboCup for the year 2050 is to compete and win against the reigning human world champions.

1.2 Goal

To achieve good performance in object segmentation, detailed labels are usually required. However, there are not always enough detailed labels available. With the help of weakly supervised learning, it is possible to use the often available coarse labels

to achieve good results in object segmentation tasks. This thesis wants to use less informative bounding box ground-truth labels, but manipulate the predictions made by the neural network during the training phase, to obtain pixel-precise predictions. The ultimate goal is to achieve competitive performance with weakly supervised learning as if fully supervised learning had been utilized to train the neural network. Without adapting the structure of the neural network, this goal is to be achieved solely by changes in the processing of the data.

The RoboCup domain was chosen for the necessary context. With the help of object detection, the position of balls, goalposts and robots can be determined. For the robots to successfully play a soccer game, they need to know about the precise ball location to bend the ball in the desired direction as well as rough locations of the goalposts. The (enemy) robots count as obstacles and need to be avoided. For balls and goalposts, precise labels can already be generated relatively easily by manipulating the bounding box labels. For robots, in contrast, the situation is more complicated. Some of the robots have very complex shapes, which cannot be easily gathered from bounding boxes by geometrical adjustments of the labels. The input of the neural network will be images of recordings of RoboCup soccer games. The prediction of the network is a heat map containing the estimated position of the respective object.

2 Related work

Traditionally, problems of computer vision have been solved by algorithms with hand-crafted features in conjunction with machine learning. Mostly based on edge detection or colour, images were checked for certain features and represented as a feature descriptor, which simplified the image down to their core. Examples for such descriptors are the Haar wavelet proposed by Alfréd Haar [Lep07], the Histogram of Oriented Gradients[DT05], Scale-invariant feature transform[Low99] or Speeded Up Robust Features[BTV06]. The SIFT algorithm, introduced by Dawid Lowe [Low99], selects key positions by applying a staged filtering approach. These keys are then used to find similar objects using a nearest-neighbour classification. This approach is particularly robust against translation, scaling, rotation and other smaller noises like blurredness. Another hand-made feature descriptor is the histograms of oriented gradients, proposed by Navneet Dalal et al. [DT05]. The image to be examined is divided into cells. Gradient directions or edge orientations are stored for each cell and are combined to a feature vector. The feature vector is fed into a linear support vector machine, which then performs the classification.

Due to the good performance of deep learning, deep learning-based methods gain more and more traction, because it is possible to extract higher-level features from a larger amount of data. The big breakthrough for such models are convolutional neural networks (CNNs). In 2012 the AlexNet neural network model, proposed by Alex Krizhevsky et al. [KSH12], has won the ImageNet competition 2012 and is considered the foundation for deep learning object recognition tasks on a larger scale. AlexNet's architecture consists of five convolutional and three fully-connected layers and was split into two pipelines. Thus the network could be spread across two different GPUs which only communicated with each other at certain layers. More recently the ResNet, proposed by Kaiming He et al. [He+16], also won the ImageNet competition 2015. The success of this network was mainly due to the introduction of residual building blocks and shortcut connections. For example, the identity function is not represented by another function, but by setting the corresponding residual to zero. It has up to

152 layers with an error rate of 3.6% on the ImageNet data set surpassing the human error rate of 5.1%[1]. This approach makes it possible to successfully train particularly large deep neural networks.

The Hamburg Bit-Bots Team is also using a CNN for the detection of balls in the RoboCup Kid-Size league [SBB18]. The CNN is based on the neural network model used in the Sweaty Robot [Sch+17], which was one of the finalists in the RoboCup Adult-Size league in 2017. In an earlier work, the CNN was also tasked to detect other objects like goalposts and robots [BG18]. However, neither the goalposts nor the robots could not be identified satisfactorily.

In recent years weakly supervised learning models are gaining more and more attention [Zho17]. Since they do not require richly labelled data, consistently successful weakly supervised learning trained networks would provide a range of new opportunities for applications with scarce data. The fact that these approaches can indeed rival their supervised learning counterparts has already been shown by several examples. On the Pascal VOC 2012 data set [Eve+], IRNet [ACK19], which is trained with coarse labels, achieves the state-of-the-art performance in both instance and semantic segmentation. For this approach pseudo instance segmentation labels are generated via Class Attention Maps. The maps are rough estimates of where a specific object class in an image is present but can't distinguish between different instances. Afterwards a known CNN model, for example, a ResNet architecture, is trained with the pseudo labels. Likewise, the Simple Does It approach [Kho+17] achieves $\sim$ 95% of the accuracy achievable with fully supervised learning approaches on using the Pascal VOC12 and COCO [Lin+14] data sets. Here, mainly recursive training is proposed, using the predictions of the neural network as input for the following epoch with few post-processing steps. According to the paper, even slight changes in the initial predictions are enough to turn the originally coarser labels into more detailed labels. A most recent state of the art semi-supervised learning model EfficientNet-L2, proposed by Xie et al. [Xie+20], employs two neural networks that train each other. Half of the training data used consists of supervised data, the other half of so-called pseudo la-

[1] `https://www.learnopencv.com/baidu-banned-from-ilsvrc-2015/`

bels. The pseudo labels get generated by using a smaller teacher model network. The training data is then fed into a larger model and trained with noise (e.g. dropout). Additionally, this approach employs model scaling for convolutional neural networks, proposed by Barret Zoph et al. [Zop+20]. With this combination, the EfficientNet-L2 achieves Top-5 accuracy of 98.7%.

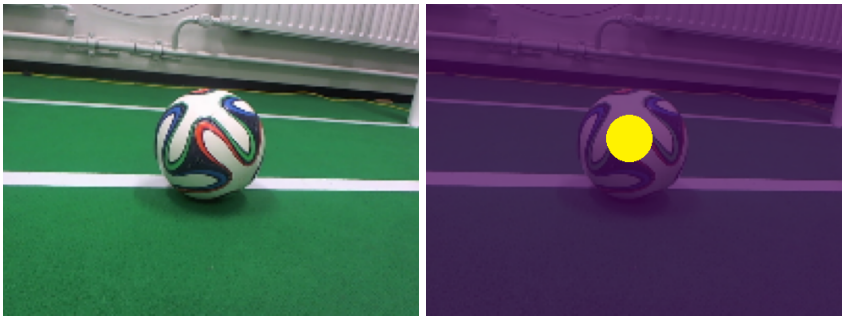
3 Background knowledge

This chapter provides the necessary basic knowledge to understand the described approaches. It includes an overview of various object detection tasks, differentiation between coarse labels and fine labels in computer vision, the concept of (convolutional) neural networks as well as information about the RoboCup.

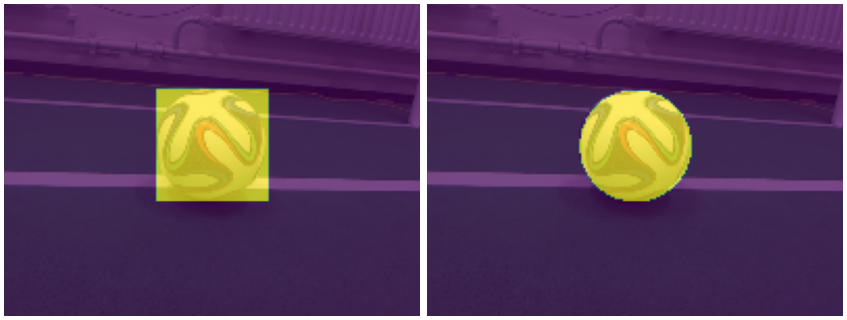
3.1 Object detection and labelling

The term object detection or object recognition can mean several different things in computer vision. While they all have in common to recognize target objects in images and videos, there are different requirements and outputs to consider. The most general is image classification. It deals with the labelling of an image according to whether it contains a specific object. This is not to be confused with object detection, where not only the classification of the image is of importance, but also the position of the desired object in the image itself. Such a position is commonly referred to as a label. There are different types of labels, which become increasingly more complex the more precise they are. An illustration of the different label types can be seen in figure 1.

- The first and simplest type of label has already been mentioned, the **classification label**. It only determines whether or not an object is visible on the image, without giving any further information about the positioning of the target object.
- The **point label** defines, as the name suggests, a point in the image that indicates the target object. The point does not necessarily have to be in the centre of the object.
- The next label type is the so-called **bounding box label**. It is a rectangular box around the target object in the image. The bounding box must contain the entire object, even if this object is partially occluded or not. However, this usually includes background pixels that do not belong to the target object.



(a) classification label: ball (b) point label



(c) bounding box label (d) pixel-precise label

Figure 1: Different types of labels shown on a ball image.

- The final, most accurate and therefore most complex label type is the **pixel-precise label**. In this case, every single pixel within an image must be checked for the object and, if necessary, labelled.

More complex labels are of course more effective for training neural networks than less accurate labels. Unfortunately, this complexity does not only occur in the level of detail of the label but also how difficult it is to obtain them. The labelling of images for a training data set is often done manually by humans. Depending on the field of application, this can only be done by professionals who have a lot of experience in the field. For example, brain tumours can only be identified in MRI scans by neurologists or otherwise qualified personnel, while the spotting of a football in an image can be done without prior knowledge in the field. On average, a classification label with one second per image and point label with 2.4 seconds per image takes very little time [Bea+16]. However, these labels are also not very informative and detail-rich. For

machine learning bounding box and pixel-precise label are used most commonly. But the time required for labelling increases rapidly the more precise the labels are. While bounding box label already take 10 seconds per image, pixel-precise label will take roughly 78 seconds per image [Bea+16]. For a data set with about 30k images pixel-precise labelling would take around 650 hours.

3.2 Neural Networks

As the term neural network suggests, such a network contains many neurons. In the classical sense, neurons are located in the brains of complex living beings like animals and humans. Every neuron is a electrically excitable cell that serves to convert signals from the body - for example what the eyes see or what the skin feels - into processable information [Wik01]. They are interconnected by synapses, which serve as a communication interface between neurons. By connecting the neurons, more complex correlations can now be drawn between the signals[Wik01]. The neurons in artificial neural networks are based on this biological brain concept [Fuk80]. In computer vision, however, these neurons are not stimulated by the signals of the body, but by images and their context and are represented using mathematical functions. An example of an artificial neural network can be seen in figure 2. Several inputs (x_i) are used so the hidden layer can make a prediction p. Each input of the hidden layer has a different weight w_i. By adding up the weights and inputs, the activation is calculated. Each neuron learns to recognize different characteristics and adjusts the internal weights accordingly. Many neurons together form a so-called layer. Every neural network must have an input and an output layer. The number of neurons in the input layer is dependent on the shape of the input data. They are there to receive the input or to make a prediction according to the learned features. Between the input and the output layer, a neural network has many hidden layers. The name comes from the fact that they are normally not accessible from outside the network [DDZ20].

3.3 Convolutional Neural Networks

Convolutional neural networks, abbreviated CNNs, are a special form of artificial neural networks, which are used commonly in the field of image processing [Guo+17]. The special feature of CNNs is the name-giving convolutional layer, which extract features from input images using convolution operations. Numerous different layers can exist within a convolutional neural network. These take over different functionalities and processing of the data. The essential layers for understanding neural network model in this thesis are explained below.

The most integral layer is the convolutional layer. This layer is the core of the network and its results have a decisive influence on the final prediction. Many of these layers exist within the network that help extract the information needed to analyze the context of the given image. When an image is first passed to a convolutional layer, each pixel and its surroundings gets filtered via several filter matrices or also called kernels. A visualization of this process can be found in the figure 3. Given the value of a pixel and the neighbouring pixels, a kernel calculates a new value and saves it in a new matrix. After each pixel and its surroundings was examined by the kernel, the output matrix is complete. This output matrix is called a feature map. Since there

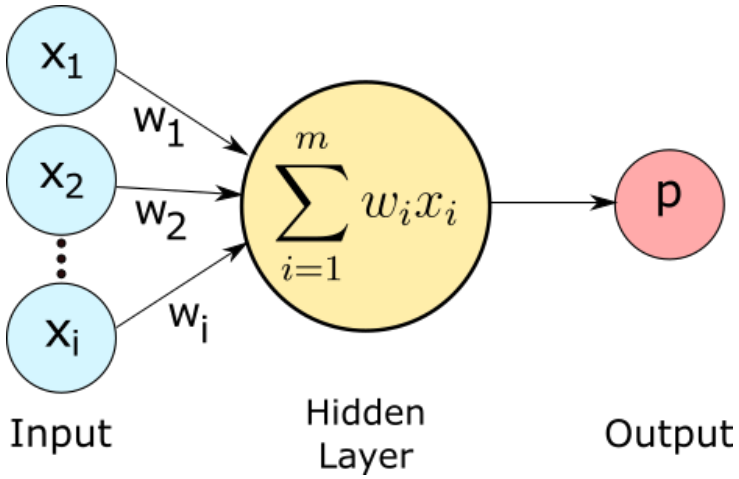


Figure 2: Example of an artificial neural network. x_i is the input, w_i denotes the weight of the connection, p is the prediction.

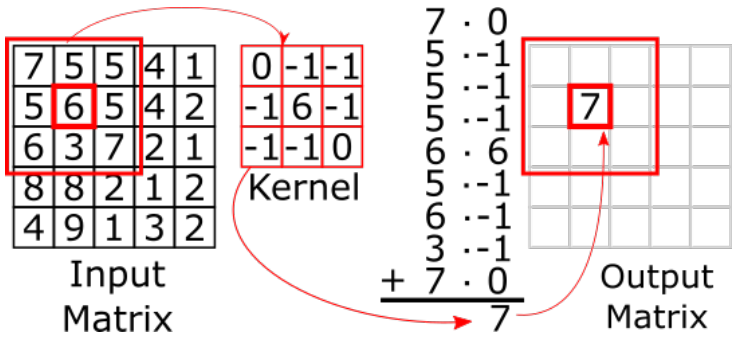
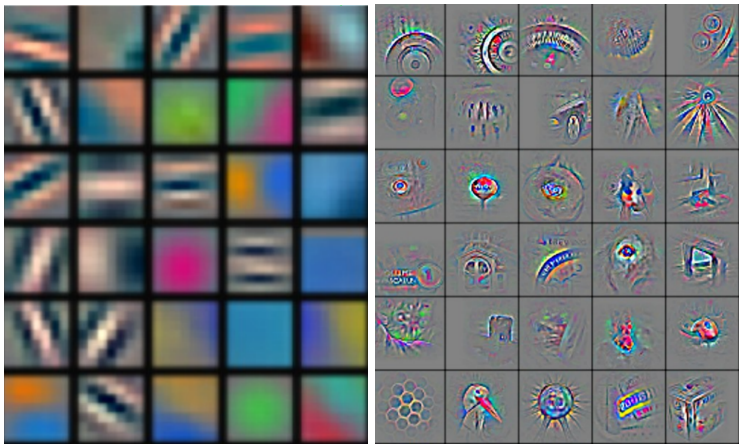


Figure 3: Functionality of a convolution layer with an incomplete output matrix.

are many kernels in one convolutional layer, several feature maps are generated per layer.

After the first feature maps have been calculated they are fed into the next convolutional layer after some minor dimensional changes done by other layers, which will be explained below. This process can be repeated as often as desired. Each convolutional layer in the network learns its kernels, which highlight specific features of an image and decreases unnecessary clutter and save its "findings" in new feature maps. As the name suggests, each feature map notes/looks for a different feature. In the first convolutional layers, these feature maps will likely look for low-level features like edges or curves in an image. At later layers, these feature maps likely contain more high-level features that are not easy to generalize. The difference between these low- and high-level feature maps is illustrated in figure 4. What exactly a CNN considers as an important feature to add to such a map is defined during training. In the last layer of a CNN, the final prediction will be made. In this thesis, the prediction of the CNN is a heat map, which predicts the location of the desired object or its possible absence in an image.

A dropout layer, which was first introduced by Nitish Srivastava et al. [Sri+14], serves mainly to counteract the overfitting of the network. The term overfitting describes the problem that arises when a network is over-trained on a specific training data set and thus achieves poorer results when later compared to the testing data set [DDZ20]. Instead of searching for robust features in the given data, the network only

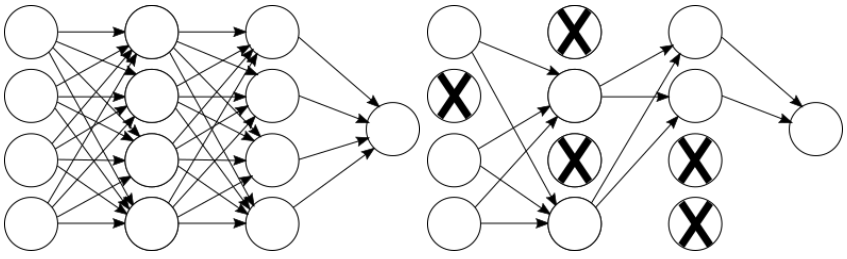


(a) low level features (b) high level features

Figure 4: Examples for a) low and b) high level features in extracted feature maps [ZF14].

learns the short cut solutions of the training data. Dropout layers simulate a kind of noise during the training process by dropping a certain percentage of the result by randomly cutting the connections for some neurons to other neurons. This process is visualized in figure 5. This forces the network to not rely on single neurons but rather whole neuron chains instead and will help to generalize the network.

The batch normalization layers, proposed by Sergey Ioffe and Christian Szegedy



(a) normal neural network (b) neural network with dropout

Figure 5: Functionality of dropout layers. a) shows a normal neural network, b) shows the same neural network but with dropout applied, graphic inspired by towardsdatascience.com[2].

[2]https://towardsdatascience.com/machine-learning-part-20-dropout-keras-layers-explained-8c9f6dc4c9ab

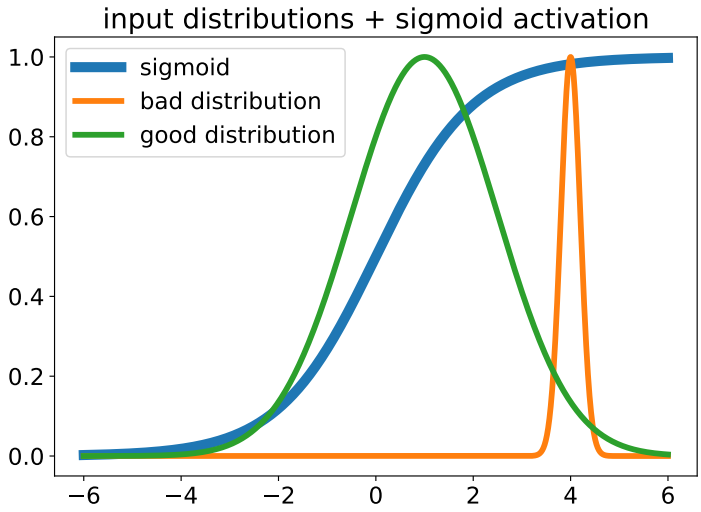
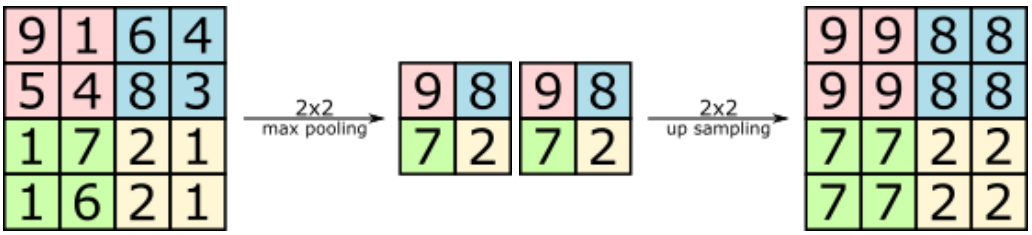


Figure 6: Examples for input distributions along a sigmoid activation function.

[IS15], are used to normalize the outputs calculated by the convolutional layers. Adjustments of the weights in the first layers of convolutional neural networks during training will affect their output and therefore the distribution of the input values for the next layer. As seen in figure 6, not all input distributions are equal. If a distribution is too narrow or shifted too far to one side, the following layer could only achieve very one-sided activations. For the here orange-coloured distribution the activation is nearly always close to 1, while the green-coloured distribution has a wide range of different activation outputs. Without the batch normalization layers, the following layers have to counteract this continuous change of the input values during training, by adapting to the changes and adjust their weights accordingly. This continuous adjustment costs time and computing power and can be reduced by normalizing said input distributions after each convolutional layer beforehand.

To reduce time complexity and needed computing power further the output of convolutional layers can be followed by a max-pooling layer. Its task is to reduce the calculated output of the previous layer by downsampling it. As a result, only important information of the input matrix is carried over and the following calculations can be performed more efficiently due to the reduced matrix dimensions without losing es-

sential information while also achieve better robustness against noise and clutter on the image [BPL10]. To reverse this effect, up sampling layers can be used to upscale individual values of the input matrix and later restore the original dimension of the matrices. Both upsampling and max-pooling layers are illustrated in figure 7.



(a) functionality of a max-pooling layer (b) functionality of an upsampling layer

Figure 7: Functionalities of a) max-pooling and b) upsampling layers.

Besides the different layers that define a neural network, several other parameters can be controlled that significantly change the behaviour. Activation functions change the way how strong and in which way each neuron in the layers is activated. Without them, neural networks would not be able to solve complex non-linear problems [DDZ20]. Common activation functions are the sigmoid function, defined in equation 1, hyperbolic tangent (tanhh), definiend in equation 2 and the rectified linear unit (ReLU), defined in equation 3. Also shown in figure 8 is a graph comparing these three activation functions to each other.

$$sig(t) = \frac{1}{1 + e^{-t}} \tag{1}$$

$$tanh(t) = \frac{e^{t} - e^{-t}}{e^{t} + e^{-t}} \tag{2}$$

$$ReLU(t) = \begin{cases} 0, & t \leq 0 \\ t, & t > 0 \end{cases} \tag{3}$$

Since the concept of neurons comes from biology and living beings learn mainly

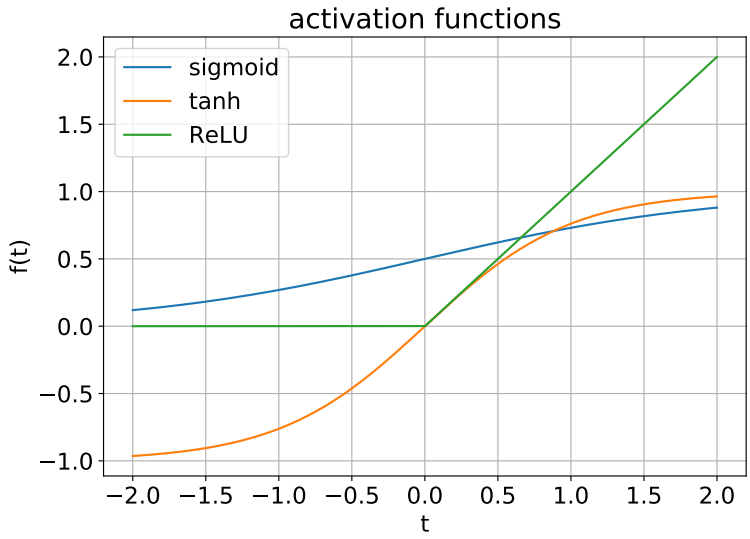


Figure 8: Graph showing common activation functions.

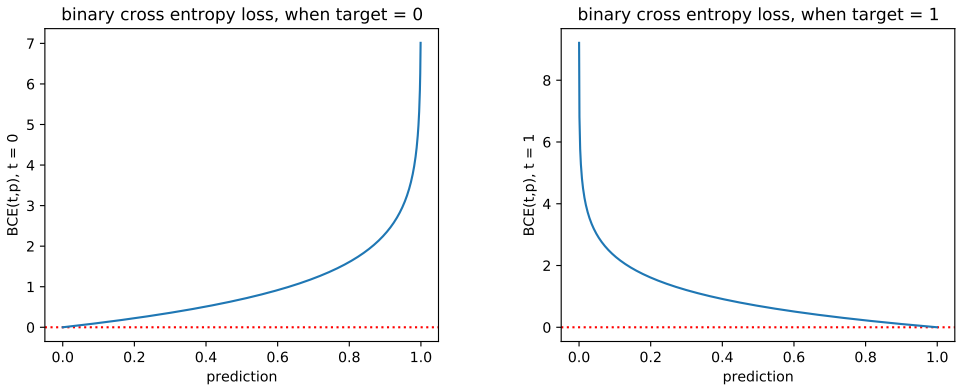
through their own mistakes, it is not surprising that artificial neural networks also learn from their mistakes. To adjust the internal weights of the neurons, an optimizer and a loss function are used. A common loss function used for classification purposes is binary cross-entropy as defined by equation 4.

$$BCE(t,p) = -(t * \log(p) + (1-t) * \log(1-p)) \tag{4}$$

The graph of the binary cross-entropy function is shown in figure 9. Using the target(t) and the prediction(p), the loss function calculates the error. The optimizer then updates the model parameters and tries to minimize this loss over the training process. In general, the more the prediction differs from the target, the greater the loss. Accordingly, the loss value is 0 if the target and prediction match.

3.4 Training

To train such a neural network for object detection purposes, different approaches can be applied. The most common type of training is supervised learning. In su-



(a) binary cross-entropy loss, t = 0 (b) binary cross-entropy loss, t = 1

Figure 9: Graph of the binary cross-entropy loss with different target values.

pervised learning, the neural network receives a picture, performs a prediction on it, and is then provided with information on how accurately the prediction was. With this information, the network and the neurons it contains can adapt and improve the predictions to achieve better results. This information is called the ground-truth. Both the prediction and the ground-truth can be any of the label types mentioned above. If both the prediction and the ground-truth are of the same label type, the approach is called supervised learning [Zho17]. In unsupervised learning, the neural network only receives the input [Gha03]. It does not receive the target output and thus does not learn based on it. The neural network is expected to search for patterns on its own, without supervised feedback.

3.5 Weakly supervised learning

The approach to training neural networks considered in this thesis is a middle ground between supervised and unsupervised learning called weakly supervised learning. There are generally three different types of weak supervision: incomplete supervision, inaccurate supervision and inexact supervision [Zho17]. With incomplete supervision, only a small amount of training data is labelled with ground-truth labels. In the case of inaccurate supervision, the ground-truth labels are not always correct. And finally,

inexact supervision is based on the fact that although the ground-truth labels are correct, they are low-level labels. For example, the neural network used in this thesis is trained via bounding box ground-truth labels but is expected to calculate pixel-precise labels for the final prediction.

3.6 RoboCup

The RoboCup, short for Robot (Soccer) World Cup, is an international competition and scientific initiative to establish a common problem (space) to promote robotics and AI research specifically. The objective, as first proposed in 1995 [Kit+95], is to have a team of fully autonomous humanoid robot soccer players compete against the winner of the most recent World Cup and win by the middle of the 21st century while complying with the official rules of FIFA - the International Federation of Association Football - which regulate the game rules across the world. The first RoboCup was held in 1997, which means the whole initiative has 50 years to accomplish this task. While it currently seems impossible for a team of robots to defeat their human counterparts in soccer, a total of 50 and at present 30 years is a long time in research and similarly impossible projects have already been achieved in the same time. About 50 years also passed between the invention of the digital computer and the development of Deep Blue, a computer that in 1996 for the first time was able to defeat the reigning world champion Garri Kasparow in chess [Rob][Sch] . In an even shorter period of additionally 20 years, Google DeepMind developed the computer program AlphaGo, which defeated the reigning world champion Fan Hui in Go in 2015[BBC].

In addition to the FIFA rules for soccer, the robots may only perform human-like functions. Since humans can only rely on their five senses, of which only hearing, seeing and playing football are essential, robots should also only act on the visual, auditory and tactile levels. This means, for example, that only two cameras per robot are allowed, based on how human vision works and that communication between robots must not be established through radio networks. They are also divided into different leagues that correspond to the sizes of the robots. While the actual aim

of the RoboCup naturally applies to the humanoid adult-size robots, research and experience on small robots are less expensive and the knowledge gained can help the initial goal of defeating the human world champions on their bigger counterparts.

The Hamburg Bit-Bots is a student workgroup for the RoboCup project of the Department of Informatics at the University of Hamburg [Bitb]. They participate in the Humanoid Kid Size League since 2012. This bachelor thesis utilizes both the data set and the basic framework of the neural network that is currently used on the robots for ball detection to analyze weakly supervised learning techniques.

4 Approach

This chapter describes the training preparations and the different experiments conducted in the thesis. After the data set and the model of the neural network are explained, the different approaches are presented. Instead of making changes to the model of the network, the main focus here is on ways of adapting the training process to achieve better results. The experiments conducted are mainly based on the Simple Does it approach [Kho+17]. All results achieved are presented and evaluated in chapter 5.

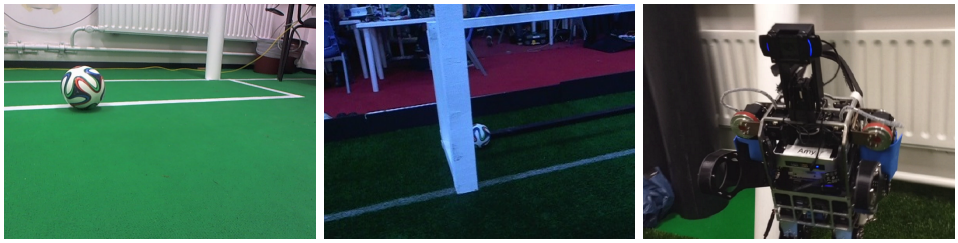
4.1 Data set

To be able to train and test a neural network (weakly) supervised, a lot of image data is necessary. Examples of the objects and important in this work are shown in the figure 10. For balls, the Hamburg Bit-Bots Team has proposed their ball Data set called Hamburg Bit-Bots Ball Data set 2018 [SBB18], which is publicly available[3]. The ImageTagger[4] [FBH18; Bita; Fie] from the Hamburg Bit-Bots Team is used as a source for the needed data of goalposts and robots. It is an open-source online platform for collaborative image labelling with the RoboCup in mind. The images were taken from the cameras that are used in the robots utilized for the RoboCup by the Hamburg Bit-Bots Team. They consist mainly of still images from video recordings gathered while the robots were playing matches of soccer. The balls on the images can lay motionless on the grass or the balls move during the recording. For all three objects, the image may also be blurred. This is due to the movement of the robots during a match.

The labels for both data sources were set manually for balls, goalposts, robots. The absolute numbers of available images in the data set are shown in the table 1. Most images were annotated with bounding box labels only, as the cost-benefit factor

[3] `https://robocup.informatik.uni-hamburg.de/en/documents/bit-bots-ball-dataset-2018/`

[4] ImageTagger `https://imagetagger.bit-bots.de`



(a) Object: ball (b) Object: goalpost (c) Object: robot

Figure 10: Examples of objects to be recognized in the RoboCup context.

is best here. For the training of weakly supervised neural networks itself, this is quite sufficient. To be able to make qualitative statements about the correctness and robustness of the experiments, later on, pixel-precise labels are essential. For ball labels, the radius is determined by half the length of the bounding box and thus the pixel-precise label is calculated via the area and radius of the ball. Since goalposts already have a rectangular shape, the bounding box labels are largely retained here. For robot labels, the task becomes more difficult. Due to their different complex shapes for arms, legs and also heads, it is not so easy to calculate from robot bounding box labels to pixel-precise labels. Therefore it is, unfortunately, necessary to go back to manual polygonal labelling for pixel-precise robot labels, which takes a lot of time. As highlighted in table 1 there are no training images and only very few test images for robots with pixel-precise labels available.

Before the images are used for training, they are scaled down to the dimension 200x150x3. The first number describes the width, the second number the height and the third number the number of colour channels of the image. According to [SBB18] this ensures a good balance between computing time and the detection rate on the robots for the calculation. Furthermore, a distinction is made between training and test/validation data sets. No image appears in both data sets simultaneously. This distinction is important, if training data were equal to test data, one could simply let the neural network train for so long that it simply learns the training data by memory. The individual neurons of the neural network no longer search for distinctive characteristics of the objects, but only learn to associate the solution to be selected with each image

- this could lead to overfitting.

An epoch describes whenever the neural network runs through the training data set one complete run. In this thesis, the different experiments are trained for 15 and 25 epochs. The test data set is run through once after the training and serves to check the learning progress and to aggregate the results for evaluation in chapter 5.

object type	# pixel-precise (training)	# bbox (training)	# pixel-precise (test)	# bbox (test)
ball	22927	22927	2177	2177
goalpost	13595	13595	2826	2826
robot	0	992	67	346

Table 1: Amount of images in training and test/validation data sets.

4.2 Base Model

Each experiment uses the same neural network architecture, which is illustrated in figure 11. This is the same architecture that is currently being used by the Hamburg Bit-Bots Team for ball detection [SBB18] with a total of 495505 trainable parameters. The structure of the network is based on the CNN used in the humanoid robot Sweaty, the finalist of RoboCup 2017 in the RoboCup Soccer Adult-Size League [Sch+17]. Due to the lower computational power of the RoboCup Humanoid Kid-Size robots of the Hamburg Bit-Bots, the input images used here are downscaled to 150x200x3, instead of the originally 640x512x3 dimensions used for the Sweaty robot. This ensures the predictions of the neural network can still be made in real-time on the robots. The scaled-down images are fed into the input layer. The various convolutional layers are responsible for feature extraction. Each convolutional layer is followed by a batch normalization layer to normalize the values during training. To ensure the images are processed in time, the inputs of the subsequent layers are further dimensionally reduced by two max-pooling layers down to a minimum shape of (38, 50). To get back to the original resolution in width and height for the final prediction, two upsampling layers are used. Furthermore, concatenations are used in certain places of the neural network. Those are meant to create connections between the simple features de-

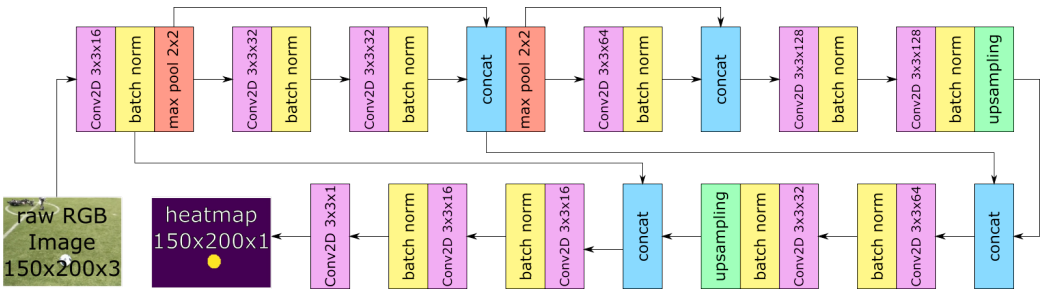


Figure 11: Base model architecture used for the experiments based on [SBB18]. Convolutional layers (purple), batch normalization (yellow), max-pooling (red), concatenation (blue), upsampling layers (green).

tected at the beginning of the network with the higher-level features of the later layers. The final prediction made by the last convolutional layer of the neural network is a heat map of the shape 150x200x1. In the heat map is the predicted location of the desired object or its possible absence in the input image.

Regarding the other parameters, the loss function used here is binary cross-entropy. As the name suggests, the neural network is thus able to distinguish between two classes - in this case, either object pixels (1) or non-object pixels (0). In other words, a probability is calculated for each image pixel between 0 and 1 whether the desired object exists. Since only these two scenarios are possible, a sigmoid activation function is used in the last convolutional layer for the final predictions. As you can see in figure 8, the input values of the sigmoid function are mapped to values from 0 to 1, which correspond to the two classes.

In the past, this specific network was only used for ball detection by the Hamburg Bit-Bots Team. Furthermore, it was also only trained fully supervised with pixel-precise labels. In a previous paper [BG18], it was examined whether the network can also recognize other objects. The results of the fully supervised trained experiment can be found in chapter 5.1.

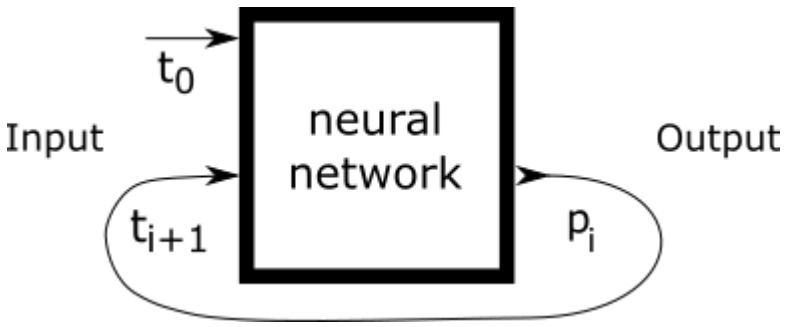


Figure 12: Recursive training approach - for the first training epoch the original ground-truth labels (t_0) will be used, afterwards, the prediction of the network (p_i) will be used as ground-truth for the next training epoch ($t_i + 1$).

4.3 Recursive Training

From this section on the neural network is trained weakly supervised. For this work, in particular, it means that pixel-precise labels are no longer used for training, but instead the less precise bounding box labels. While the structure of the neural network is retained without changes, the input of the network will be modified. Based on the Simple Does It paper [Kho+17] mentioned in chapter 2 the approach of recursive training is used here.

For the first training epoch, the original bounding box labels are still used. After the first epoch, the predictions made by the network are available. These previous predictions are now used as the new ground-truth labels without any adaptations for the next training epoch. This process is repeated until the desired number of training epochs is reached. An illustration of the process is shown in figure 12.

4.3.1 Recursive Training - box

The previously presented naive approach has some inherent problems. The first predictions may be entirely wrong. The network might return empty labels even though an object is present on the image, or it might mark the whole image as an object. If the network continues to train with this prediction, it would not achieve good results.

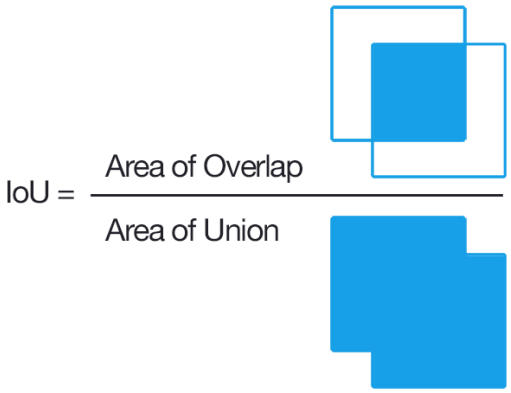


Figure 13: Equation of Intersection over Union[5].

However, the original bounding box labels can be used to tweak the predictions. All pixels outside the bounding box are guaranteed not to be object pixels. So if the predictions claim that objects are outside the bounding box, such pixels will be reset to zero.

4.3.2 Recursive Training - box + intersection over union

The box approach has addressed how the original labels can be used outside the bounding box to improve predictions. However, there may still be poor annotations within the bounding box of the new predictions. With the help of the intersection over union metric, an additional rule for resetting the original label is considered. Even though intersection over union will be further explained in chapter 5, a visual description of the metric is shown in figure 13.

If 50% or more of the prediction matches the original bounding box label, the prediction is assumed to be good. This prediction is then used for the next training epoch. If however, the value is below 50%, the original label is used as the ground-truth instead of the prediction for the next training epoch. This also intercepts the case what happens when the network returns completely blank labels as prediction although the

[5] https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

object is present in the image. If these were used unchanged as ground-truth for the next epoch for training, the results would probably be poor. Since completely missing labels are worse than the original bounding boxes, the bounding boxes are restored in this case. An example of how the box + intersection over union approach modifies the prediction labels is shown in picture 14.

(a) prediction (b) original bbox label (c) modified label

Figure 14: Example for the box + intersection over union method. a) the prediction made by the neural network with a lot of unwanted background, b) the original bounding box label, c) the modified prediction used for the next training epoch.

4.4 GrabCut

In addition to the recursive training approach discussed above, another possibility of manipulating the data will be discussed here. The GrabCut algorithm, proposed by Rother et al. in 2004, [RKB04], is an interactive way to separate objects from the background in images. The algorithm distinguishes between definite background, definite foreground and unknown pixels, which can be either foreground or background. To achieve this, the graph cut algorithm is used iteratively, which originally solved the min-cut problem in graph theory [BJ01]. In this case, the graphs to be cut are the background and the target object. The function to be minimised is defined in such a way that a good separation is possible. Neighbouring pixels with similar colouring should have lower costs because they probably belong to the same category. The input of the algorithm is the original images and the information about bounding boxes and thus the positions of the target objects. All pixels outside the bounding box are

treated as definite background, while in the first iteration of the algorithm those inside the bounding box are treated as unknown. With a higher number of iterations, this result is refined. In this work, the algorithm was executed five times. The output of the algorithm is a cutout of the image with the object. With the help of the cutout, a mask is created with which the label is generated. This process is shown in figure 15. The labels generated in this way are then used to train the neural network. The GrabCut implementation of the Open Source Computer Vision Library[6] was used.

(a) original (b) cutout (c) label

Figure 15: Example for the GrabCut approach. a) original image, b) cutout of the robot object after grabcutting, c) the label used for training.

[6] https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut.html

5 Experiment result

This chapter evaluates the approaches proposed in chapter 4. To compare the different experiments with each other, the results are visualized and compared based on the accuracy, precision, recall and the intersection over union [OD08]. These metrics used are also explained below. To shorten the formulas of the metrics, the respective abbreviations used stand for TP: *True Positive*, FP: *False Positive*, TN: *True Negative* and FN: *False Negative*. The prefixes represent the correctness of the statement, while the suffixes represent *positive* for object pixel found and *negative* for non-object pixel.

- **Accuracy**: Describes the ratio between correctly recognized object and areas correctly recognized as non-object divided by all results.

$$\frac{TP + TN}{TP + FP + FN + TN} = accuracy \tag{5}$$

- **Precision**: This value expresses how many as true predicted object pixels are in fact correct. If the value is high, it is highly probable that all areas labelled by the neural network as true are very likely recognizable objects.

$$\frac{TP}{TP + FP} = precision \tag{6}$$

- **Recall**: Reflects the sensitivity of the results and indicates how many of all pixels to be detected the network has recognised.

$$\frac{TP}{TP + FN} = recall \tag{7}$$

- **Intersection over Union**: The Intersection over Union *(IoU)*, is the most important measurement for this thesis. It describes how exactly the predicted labels fit into the correct ground-truth labels. The better the calculated labels correspond to the ground-truth labels, the higher the IoU. On the other hand, the more the

calculated labels differ from the ground-truth labels, the lower this value and the worse the prediction. This metric is particularly interesting because a high value reflects exactly the behaviour that is desirable from the training of the neural network.

$$\frac{TP}{TP + FP + FN} = Intersection\ over\ Union \tag{8}$$

All metrics are rounded to the first decimal place and are also given in percentiles. Percentiles indicate the values below which a certain percentage of the data in a data set is found. This is important because it allows outliers in the training and test data to be accommodated for. Additionally, the calculated predictions are compared to pixel-precise labels, as these would be the desired result.

5.1 Base Model

To be able to compare the approaches, the network was trained fully supervised with pixel-precise training data. The results are shown in table 2. Furthermore, exemplary predictions for each object are shown in figure 16. The outcome for the object class ball is promising as expected. With $\sim$71% mean IoU the network detects most parts of the ball. From the precision and recall it can also be seen that there were both incorrectly predicted object pixels and that not all object pixels were found by the neural network. For the goalposts and robots, in contrast, the network could not achieve good results. While some parts of goalposts were at least recognised, they were never fully detected. Almost no robot was correctly recognized. The generally high accuracy value for all target objects is due to the comparatively small amount of target object in the image compared to the background. For goalposts and also robots, the neural network seems to prefer to declare the entire image as background pixels, instead of searching for the actual objects.

For comparison, table 3 shows the training with the less detailed bounding box labels. For the balls, the value for precision and the IoU decreases by about 8%. Since

base model - pixel-precise label

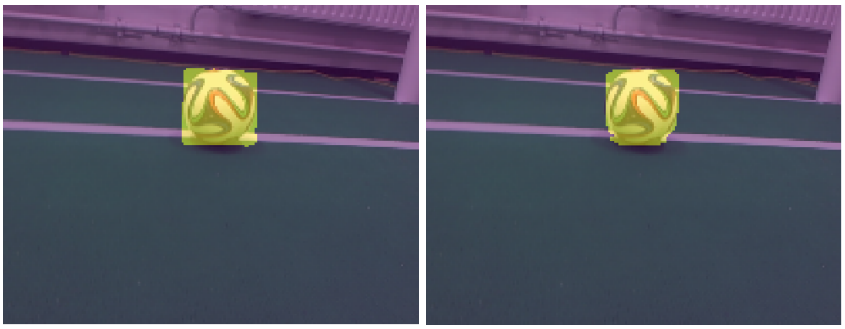
object	ball		goalpost		robot	
# training epochs	15	25	15	25	15	25
accuracy, mean	0.9988	0.9989	0.9921	0.992	0.9992	0.9941
accuracy, P_{90}	0.9998	1.0	1.0	1.0	1.0	0.9996
accuracy, P_{99}	0.9999	1.0	1.0	1.0	1.0	1.0
precision, mean	0.867	0.8632	0.0675	0.066	0.0	0.0119
precision, P_{90}	1.0	1.0	0.0	0.0	0.0	0.0
precision, P_{99}	1.0	1.0	1.0	0.7191	0.0	0.4832
recall, mean	0.8151	0.7842	0.0149	0.0272	0.0	0.0031
recall, P_{90}	1.0	1.0	0.0	0.0	0.0	0.0
recall, P_{99}	1.0	1.0	0.4002	0.7191	0.0	0.1335
IoU, mean	0.7542	0.7124	0.0143	0.0245	0.0	0.0264
IoU, P_{90}	0.9416	0.9359	0.0	0.0	0.0	0.0
IoU, P_{99}	0.9867	0.9914	0.3883	0.6198	0.0	0.1176

Table 2: Results of training the base model fully supervised with pixel-precise labels. Mean denotes the mean value, P_X denotes the Xth-percentile value.

(a) ball (b) goalpost (c) robot

Figure 16: Example predictions for each object class with fully supervised training. Yellow patches denote the predictions of the network, red bounding boxes declare the not detected objects. a) correctly identified ball, b) not detected goalposts, c) wrongly detected robots.

we now train with the less rich bounding box labels, the neural network also identifies some pixels around the ball as object pixels, although these are background. This can be seen from the poorer values in precision. Only the recall increases compared to the pixel-precise training. The accuracy remains similar, probably due to the amount of background in the image compared to the size of the target object ball. As was previously the case with the pixel-precise labels, the goalposts and robots could not achieve any satisfactory results. Although the values for accuracy are very high, the other three metrics reveal that this is also only in the correctly recognised background. Thus the network simply marks each presented image as background rather than searching for the object.



(a) none (b) naive

(c) box (d) box + iou

Figure 17: Predictions of the various recursive training iterations shown on a ball image. It can be seen that each refinement can lead to a small improvement of the predictions.

5.2 Recursive Training

The results for the naive approach are shown in table 4 can are visualised using predictions in figure 17. As a reminder, the predictions of the previous epoch were used here unaltered as ground-truth for the next epoch. For balls, the values in the IoU have now fallen considerably. On average, only about half of the pixels are now correctly recognised. Especially the recall has decreased because now many object pixels are not detected. As before, the other two target objects are not detected at all.

The results for the box approach are shown in table 5. This approach is intended to reset the pixels of the recursive training that are outside the bounding box to background. Compared with the values of the naive approach, the values have already improved significantly for the target class ball. Instead of the 53,4% in the mean IoU the value is up to 63,95% for 25 epochs training. The values for Recall and Precision also increased slightly. But even after 15 epochs, improvements compared to the naive approach can be seen. The values of the other two target objects are still poor.

The results of the box + intersection over union approach are shown in table 6. In addition to the previously made adjustments outside the bounding box, the predictions inside the bounding box are also manipulated. Compared with the box approach, smaller improvements in the values can be seen again. But more importantly, the values are now comparable to the pixel-precise training values of table 2 for the target object ball. The mean IoU for 25 epoch training is higher than the value of fully supervised training with pixel-precise ground-truth labels. Also, the value of 75% after 15 training epochs for the fully supervised learned approach is comparatively close to the value of 73% for this method in the mean IoU. Although the changes in the values are tiny, now for the first time with recursive training even a few goalposts have been recognised, at least to some extent. Those values and the results for the robots still remain poor.

base model - bbox label

object	ball		goalpost		robot	
# training epochs	15	25	15	25	10	25
accuracy, mean	0.9986	0.9987	0.9921	0.992	0.9979	0.9992
accuracy, P_{90}	0.9996	0.9997	1.0	1.0	1.0	1.0
accuracy, P_{99}	0.9998	0.9998	1.0	1.0	1.0	1.0
precision, mean	0.7609	0.77	0.0675	0.066	0.0007	0.0
precision, P_{90}	0.9048	0.9366	0.0	0.0	0.0	0.0
precision, P_{99}	1.0	1.0	1.0	1.0	0.0	0.0
recall, mean	0.8755	0.8235	0.0149	0.0272	0.0	0.0
recall, P_{90}	1.0	1.0	0.0	0.0	0.0	0.0
recall, P_{99}	1.0	1.0	0.4002	0.7191	0.0	0.0
IoU, mean	0.7158	0.7098	0.0145	0.0244	0.0	0.0
IoU, P_{90}	0.8519	0.863	0.0	0.0	0.0	0.0
IoU, P_{99}	0.8945	0.9034	0.3883	0.6198	0.0	0.0

Table 3: Results of training the base model with bounding box labels.

Recursive training: naive - bbox label

object	ball		goalpost		robot	
# training epochs	15	25	15	25	15	25
accuracy, mean	0.9979	0.9984	0.9924	0.9924	0.1355	0.9992
accuracy, P_{90}	0.9996	0.9996	1.0	1.0	0.1731	1.0
accuracy, P_{99}	0.9998	0.9998	1.0	1.0	0.2638	1.0
precision, mean	0.7342	0.7648	0.0	0.0	0.0009	0.0
precision, P_{90}	1	1.0	0.0	0.0	0.0	0.0
precision, P_{99}	1.0	1.0	0.0	0.0	0.0272	0.0
recall, mean	0.5193	0.562	0.0	0.0	0.0332	0.0
recall, P_{90}	0.9063	0.9433	0.0	0.0	0.0	0.0
recall, P_{99}	1.0	1.0	0.0	0.0	1.0	0.0
IoU, mean	0.4962	0.534	0.0	0.0	0.0009	0.0
IoU, P_{90}	0.8361	0.8678	0.0	0.0	0.0	0.0
IoU, P_{99}	0.8893	0.9072	0.0	0.0	0.0272	0.0

Table 4: Results of training with the naive approach and bbox labels.

Recursive training: box - bbox label

object	ball		goalpost		robot	
# training epochs	15	25	15	25	15	25
accuracy, mean	0.9986	0.9986	0.9831	0.9924	0.9907	0.9991
accuracy, P_{90}	0.9996	0.9997	1.0	1.0	0.9966	1.0
accuracy, P_{99}	0.9998	0.9999	1.0	1.0	0.9995	1.0
precision, mean	0.8195	0.8237	0.0001	0.0	0.0021	0.0027
precision, P_{90}	1.0	1.0	0.0	0.0	0.0	0.0
precision, P_{99}	1.0	1.0	0.0	0.0	0.0324	0.0
recall, mean	0.6362	0.6712	0.0	0.0	0.0009	0.0002
recall, P_{90}	0.9517	0.9343	0.0	0.0	0.0	0.0
recall, P_{99}	1.0	1.0	0.0	0.0	0.0126	0.0
IoU, mean	0.5998	0.6395	0.0	0.0	0.0006	0.0002
IoU, P_{90}	0.8596	0.8636	0.0	0.0	0.0	0.0
IoU, P_{99}	0.9072	0.918	0.0	0.0	0.0095	0.0

Table 5: Results of training with the box approach and bbox labels.

Recursive training: box + iou - bbox label

object	ball		goalpost		robot	
# training epochs	15	25	15	25	15	25
accuracy, mean	0.9987	0.9987	0.9904	0.9906	0.5489	0.7081
accuracy, P_{90}	0.9997	0.9997	1.0	1.0	0.6874	0.8532
accuracy, P_{99}	0.9999	0.9998	1.0	1.0	0.7871	0.9143
precision, mean	0.8790	0.8911	0.0246	0.0687	0.0001	0.009
precision, P_{90}	1.0	1.0	0.0	0.0	0.0	0.0
precision, P_{99}	1.0	1.0	1.0	1.0	0.0	0.028
recall, mean	0.7449	0.756	0.0018	0.0083	0.0015	0.0037
recall, P_{90}	0.9333	0.9211	0.0	0.0	0.0	0.0
recall, P_{99}	1.0	0.967	0.0492	0.2179	0.0	0.1387
IoU, mean	0.71	0.7332	0.0014	0.0069	0.0001	0.0007
IoU, P_{90}	0.8672	0.8876	0.0	0.0	0.0	0.0
IoU, P_{99}	0.9278	0.9362	0.0362	0.1938	0.0	0.0226

Table 6: Results of training with the box + intersection over union approach and bbox labels.

5.3 GrabCut

The results of the GrabCut approach can be seen in table 8. Compared to the values from table 3 where the bounding box label without GrabCut was used, the values for this approach are unfortunately worse. The mean IoU falls by 10%. The values for precision and recall have also decreased. Example predictions of this method for the target class ball are shown in figure 18. The GrabCut approach had problems with partially obscured objects, especially because the algorithm could not distinguish between foreground and background. In general, good results were achieved when the ball was in the open. Only the value for the accuracy is high, which is still due to the ratio between object and background. The values of the other two target objects remain poor. Noticeable, however, is the comparatively even worse accuracy value for 25 training epochs for robots.

5.4 Comparison

In this section, the different approaches are compared to each other. Since the results for the object classes goalpost and robot are not meaningful for analysis, they are excluded from this comparison. The table 7 shows the best results for each method. The entries highlighted in yellow are the fully supervised trained neural network results.

comparison

metric	accuracy	precisicion	recall	IoU
fully supervised pixel-precise	0.9988	0.867	0.8151	0.7542
fully supervised bbox	0.9986	0.7609	0.8755	0.7158
Recursive training: naive	0.9984	0.7648	0.562	0.534
Recursive training: box	0.9986	0.8237	0.6712	0.6395
Recursive training: box + iou	0.9987	0.8911	0.756	0.7332
GrabCut	0.9983	0.8737	0.6265	0.6079

Table 7: Best results of each method. For each metric the mean value is shown. Yellow shaded entries are fully supervised trained, the green shaded entry is the best weakly supervised trained approach.

The best possible method is, of course, the fully supervised learned neural network using pixel-precise data for training. The values of this approach should be pursued by other approaches. In particular, the mean IoU value of 75.42% should be met. By far the worst approach was the naive recursive training. The mean IoU here is only 53.4%, which is around 20% worse than the aimed-for value. Merely using the network's unmodified predictions for training has significantly worsened the performance of the network. Furthermore, the GrabCut approach as well as the box iteration of the recursive training are not better than the fully supervised training with bounding box labels with values below 71,58% in the mean IoU. The best approach of the weakly supervised methods tested here is the box + intersection over union approach of the recursive training with a mean IoU of 73,32%. This approach is also better than the fully supervised training with bounding box labels. Additionally, this method is just 2% worse than fully supervised trained neural networks with pixel-precise training data. The mean precision value of 89,11% is higher than the mean precision 86,7% of the pixel-precise training method, which indicates that the method is slightly more robust. If object pixels are found in the image, they are more likely to be correct than those of the pixel-precise trained neural network. The lower value of the mean recall does indicate, however, that the neural network does find fewer object pixels in general.

GrabCut - bbox label

object	ball		goalpost		robot	
# training epochs	15	25	15	25	15	25
accuracy, mean	0.9983	0.9985	0.9874	0.9897	0.994	0.6611
accuracy, P_{90}	0.9996	0.9996	0.9946	0.9812	0.9984	0.7587
accuracy, P_{99}	0.9998	0.9997	1.0	1.0	0.9993	0.8239
precision, mean	0.8737	0.8620	0.0013	0.0024	0.0009	0.0004
precision, P_{90}	1.0	1.0	0.0	0.0	0.0	0.0
precision, P_{99}	1.0	1.0	0.042	0.0421	0.0303	0.0139
recall, mean	0.6265	0.5946	0.0003	0.0004	0.0002	0.0033
recall, P_{90}	0.8622	0.8559	0.0	0.0	0.0	0.0
recall, P_{99}	0.9189	0.9155	0.0054	0.0064	0.0065	0.1177
IoU, mean	0.6079	0.5872	0.0004	0.0006	0.0002	0.0003
IoU, P_{90}	0.8557	0.8486	0.0	0.0	0.0	0.0
IoU, P_{99}	0.9057	0.9035	0.0071	0.0084	0.0055	0.0127

Table 8: Results of training with the GrabCut approach and bbox labels.

(a) good result (b) bad result with partially obscured ball

Figure 18: Examples for predictions with the GrabCut approach.

6 Conclusion

In this thesis, weakly supervised learning methods that had already been successfully applied in the Simple does it approach were tried out in the RoboCup context. Good performance was achieved for the final iteration of the recursive training method in particular. The results of the target class ball and the box + intersection over union approach were only ~ 2 - 3 % behind the results of the model trained in a supervised manner with pixel-precise labels. Although the results of the other two target classes goalposts and robots are inferior with the same approach, they are also poor for the fully supervised learned model. To detect these objects as well as the target class ball, the changes made in the experiments of this thesis to the training data alone are not sufficient. This would probably require additional changes to the neural network structure or an entirely different model.

In this work, the GrabCut algorithm is used to highlight the object independently. Unfortunately, the GrabCut approach was not successful. Usually, at least a small amount of user input is expected and advised to refine and correct the cutouts made by the algorithm. By operating without user interaction, the cutouts may contain too much or less target object than desired. Thus the execution of the algorithm alone does not lead to satisfactory results.

In total the experiments show, it is at least possible to achieve competitive results with weakly supervised learning methods and coarse labels. Using weakly supervised methods in RoboCup scenarios also shows that this success is not purely context-dependent. The methods also work with other data sets, like the Hamburg Bit-Bots Ball Data set 2018 used in this thesis, and not only with the popular data sets like Pascal VOC 2012 or COCO as tested in the Simple Does It paper.

7 References

[ACK19] Jiwoon Ahn, Sunghyun Cho, and Suha Kwak. “Weakly supervised learning of instance segmentation with inter-pixel relations”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2209–2218.

[BBC] BBC. *BBC, Artificial intelligence: Google’s AlphaGo beats Go master Lee Se-dol*. https://www.bbc.com/news/technology-35785875. Online; accessed 13-July-2020.

[BBU18] Hakan Bilen, Rodrigo Benenson, and Jasper Uijlings. *Weakly Supervised Learning for Computer Vision. CVPR 2018 Tutorial*. Online; accessed 5-February-2020. June 18, 2018. URL: http://web.inf.ed.ac.uk/ipab/news/cvpr-tutorial-2018-hakan-bilen.

[Bea+16] Amy Bearman et al. “What’s the point: Semantic segmentation with point supervision”. In: *European conference on computer vision*. Springer. 2016, pp. 549–565.

[BG18] Maximilian Birkenhagen and Robert Geislinger. “Ball-, Torpfosten- und Roboter-Erkennung im RoboCup durch FCNNs”. In: 2018.

[Bita] Hamburg Bit-Bots. *Github - ImageTagger*. Online; accessed 13-October-2019. URL: https://github.com/bit-bots/imagetagger.

[Bitb] Hamburg Bit-Bots. *Hamburg Bit-Bots, Welcome!* https://robocup.informatik.uni-hamburg.de/en/welcome/. Online; accessed 13-July-2020.

[BJ01] Yuri Y Boykov and M-P Jolly. “Interactive graph cuts for optimal boundary & region segmentation of objects in ND images”. In: *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*. Vol. 1. IEEE. 2001, pp. 105–112.

[Boj+16] Mariusz Bojarski et al. “End to end learning for self-driving cars”. In: *arXiv preprint arXiv:1604.07316* (2016).

[BPL10] Y-Lan Boureau, Jean Ponce, and Yann LeCun. “A theoretical analysis of feature pooling in visual recognition”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 111–118.

[BTV06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “Surf: Speeded up robust features”. In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[Che+18] Liang-Chieh Chen et al. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 801–818.

[DDZ20] Hao Dong, Zihan Ding, and Shanghang Zhang. *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Springer Nature, 2020.

[DT05] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*. Vol. 1. IEEE. 2005, pp. 886–893.

[Eve+] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[FBH18] Niklas Fiedler, Marc Bestmann, and Norman Hendrich. “Imagetagger: An open source online platform for collaborative image labeling”. In: *Robot World Cup*. Springer. 2018, pp. 162–169.

[Fed19] RoboCup Federation. *RoboCup Home*. Online; accessed 13-July-2020. June 15, 2019. URL: https://www.robocup.org/.

[Fie] Niklas Fiedler. *Imagetagger: An Open Source Online Platform for Collaborative Image Labeling*. poster.

[Fuk80] Kunihiko Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological cybernetics* 36.4 (1980), pp. 193–202.

[Gha03] Zoubin Ghahramani. “Unsupervised learning”. In: *Summer School on Machine Learning*. Springer. 2003, pp. 72–112.

[Guo+17] Tianmei Guo et al. “Simple convolutional neural network on image classification”. In: *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(*. IEEE. 2017, pp. 721–724.

[Hal+20] David Hall et al. “Probabilistic object detection: Definition and evaluation”. In: *The IEEE Winter Conference on Applications of Computer Vision*. 2020, pp. 1031–1040.

[He+16] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[Hua96] T Huang. *Computer vision: Evolution and promise*. 1996.

[IS15] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[Kho+17] Anna Khoreva et al. “Simple does it: Weakly supervised instance and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 876–885.

[Kit+95] Hiroaki Kitano et al. *RoboCup: The Robot World Cup Initiative*. 1995.

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

[Lep07] Ülo Lepik. “Application of the Haar wavelet transform to solving integral and differential equations.” In: *Proceedings of the Estonian Academy of Sciences, Physics, Mathematics*. Vol. 56. 1. 2007.

[Lin+14] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2014. arXiv: `1405.0312 [cs.CV]`.

[Low99] David G Lowe. “Object recognition from local scale-invariant features”.

In: *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[OD08] David L Olson and Dursun Delen.
Advanced data mining techniques.
Springer Science & Business Media, 2008.

[Pap+15] George Papandreou et al.
“Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1742–1750.

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake.
“" GrabCut" interactive foreground extraction using iterated graph cuts”. In: *ACM transactions on graphics (TOG)* 23.3 (2004), pp. 309–314.

[Rob] RoboCup. *RoboCup, Objective*.
`https://www.robocup.org/objective`.
Online; accessed 13-July-2020.

[SBB18] Daniel Speck, Marc Bestmann, and Pablo Barros.
“Towards real-time ball localization using cnns”.
In: *Robot World Cup*. Springer. 2018, pp. 337–348.

[Sch] André Schulz. *Chessbase, 20 Jahre Kasparov gegen Deep Blue*.
`https://de.chessbase.com/post/20-jahre-kasparov-gegen-deep-blue`. Online; accessed 13-July-2020.

[Sch+17] Fabian Schnekenburger et al.
“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. sn. 2017.

[Shi+16] Hoo-Chang Shin et al. “Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning”. In: *IEEE transactions on medical imaging* 35.5 (2016), pp. 1285–1298.

[Sri+14] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[Wik01] Wikipedia contributors.
Neuron — Wikipedia, The Free Encyclopedia.
[Online; accessed 28-May-2020]. 2001.
URL: https://en.wikipedia.org/wiki/Neuron.

[Xie+20] Qizhe Xie et al.
“Self-training with noisy student improves imagenet classification”.
In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10687–10698.

[ZF14] Matthew D Zeiler and Rob Fergus.
“Visualizing and understanding convolutional networks”.
In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[Zho17] Zhi-Hua Zhou.
“A brief introduction to weakly supervised learning”.
In: *National Science Review* 5.1 (2017), pp. 44–53.

[Zop+20] Barret Zoph et al. “Rethinking pre-training and self-training”.
In: *arXiv preprint arXiv:2006.06882* (2020).

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.