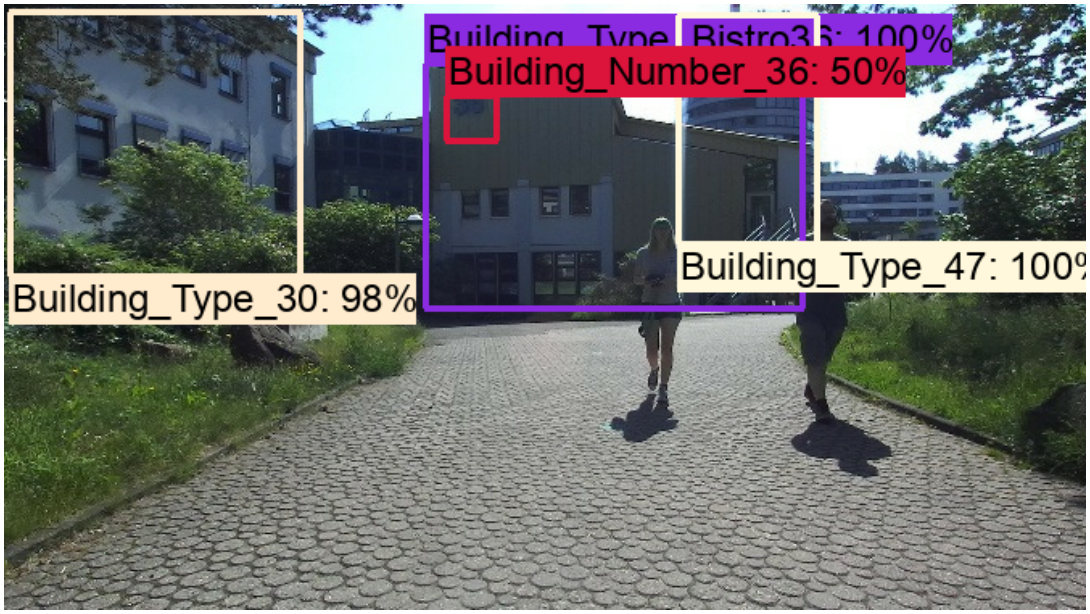
Robotics Research Lab
Department of Computer Science
Technische Universität Kaiserslautern

Bachelorarbeit



Object Detection for Localization in a Campus Environment

Selina Heller

January 17, 2022

Bachelorarbeit

Object Detection for Localization in a Campus Environment

Robotics Research Lab
Department of Computer Science
Technische Universität Kaiserslautern

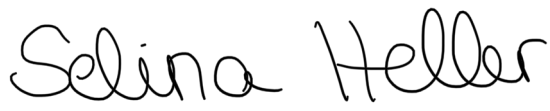
Selina Heller

Day of issue : 01.07.2021
Day of release : 17.01.2022

First Reviewer : Prof. Dr. Karsten Berns
Second Reviewer : Axel Vierling
Supervisor : Axel Vierling

Hereby I declare that I have self-dependently composed the Bachelorarbeit at hand. The sources and additives used have been marked in the text and are exhaustively given in the bibliography.

January 17, 2022 – Kaiserslautern



(Selina Heller)

Abstract

This thesis with the title "Object Detection for Localization in a Campus Environment" presents a custom object detector to complement the localization of vehicles using the Global Navigation Satellite System (GNSS) and OpenStreetMap data. The detector is based on Faster R-CNN, ResNet, and a pre-trained model provided by the TensorFlow Object Detection API. For the task of localization, a set of landmarks on the campus of the TU Kaiserslautern is identified. These are then divided into classes to create a dataset with which the network is trained. The dataset consists of images, which are generated by the Delivery-Robot of the RRLAB driving two routes on the campus. The network is trained to identify 20 different landmarks, which include buildings, artworks, benches, entrances, fire hydrants, fitness equipment, and waste bins. The goal is to evaluate the performance of the model and the applicability of the model for localization. To achieve this the model is tested on FINROC and integrated into the existing localization algorithm of Maximilian Kunz of the RRLAB. At last, an outlook on the possible improvements and additions is given.

Zusammenfassung

Diese Arbeit mit dem Titel "Object Detection for Localization in a Campus Environment" stellt einen maßgeschneiderten Objektdetektor vor, der die Lokalisierung von Fahrzeugen mit Hilfe des Global Navigation Satellite System (GNSS) und OpenStreetMap-Daten ergänzt. Der Detektor basiert auf Faster R-CNN, ResNet und einem vortrainierten Modell, das von der TensorFlow Object Detection API bereitgestellt wird. Für die Aufgabe der Lokalisierung wird eine Reihe von Landmarken auf dem Campus der TU Kaiserslautern identifiziert. Diese werden dann in Klassen eingeteilt, um einen Datensatz zu erstellen, mit dem das Netzwerk trainiert wird. Der Datensatz besteht aus Bildern, die vom Delivery-Robot des RRLAB beim Abfahren von zwei Routen auf dem Campus erzeugt werden. Das Netzwerk wird darauf trainiert, 20 verschiedene Landmarken zu erkennen, darunter Gebäude, Kunstwerke, Bänke, Eingänge, Hydranten, Fitnessgeräte und Mülleimer. Ziel ist es, die Leistung des Modells und die Anwendbarkeit des Modells für die Lokalisierung zu bewerten. Zu diesem Zweck wird das Modell auf FINROC getestet und in den bestehenden Lokalisierungsalgorithmus von Maximilian Kunz vom RRLAB integriert. Abschließend wird ein Ausblick auf mögliche Verbesserungen und Ergänzungen gegeben.

Contents

1. Introduction

1.1 Motivation

With the emergence of autonomous driving, there are many different proposed ideas to solve the problem of driving in urban spaces and in the case of this thesis on a university campus. While many researchers focus on the development of self-driving cars on the road, there are many other application areas for autonomous vehicles. They can be used for search and rescue missions as well as agriculture and in the case of this thesis in logistics. A robot delivering various things completely autonomous comes with other challenges than driving on a road since the area is often residential and without much traffic containing other vehicles. The problem of the search for a safe route to drive on is amplified by the possibility of stairs and other for the robot impassable stretches. This means that accurate localization of the robot without the help of road markings should be realized to minimize the possibilities of an accident if turning on the spot is not possible to navigate out of a situation that could lead to the robot getting stuck. Most commonly used for outdoor localization is the Global Navigation Satellite System (GNSS) to measure the coordinates of the position of the robot and match this position with a map stored in the database of the robot. This position matching is then used to correct the position of the robot if it is deviating from the position that the robot should be in. While this approach often works well in rural areas, where nothing disturbs the signal but a cloudy sky, it does usually not work great in urban areas with buildings and tunnels. Since the Delivery-Robot [PostBot 21] should drive on the campus of a university, which contains high buildings and other signal disturbances, a reliable localization only using GNSS can not be guaranteed. In [Kunz 21] a localization method is proposed, which depends on the detection of a few objects in the vicinity of the robot and the comparison of the detected objects to a high-level featuring map provided by OpenStreetMap (OSM). These objects are the content of this thesis since it deals with the detection of landmarks, which can then be used for the above described localization of the robot.

1.2 Task

In this thesis, a custom object detector is developed to detect landmarks on the campus of the TU Kaiserslautern. The dataset is produced with the help of the Delivery-Robot [PostBot 21] and labeled manually. It is investigated, if these landmarks, like buildings and their entrances and numbers, benches, wastebaskets, and fire hydrants, can be successfully detected and be used for the localization of the robot driving on the campus. To achieve this Faster R-CNN is used, utilizing a pre-trained model and ResNet as the backbone, to generate a 2D machine learning detection giving a class tag and confidence for the detection and a bounding box. This can then be converted to a 3D detection based on a point cloud, camera frustum, and the image to get the local position, distance, and angle. The final model to be used in location approaches like [Kunz 21] is saved as a frozen graph and classifies the landmarks in each frame of a video or image.

1.3 Overview

The thesis is divided into 6 chapters.

Chapter 2 explains the theoretical foundations and basics. Starting with machine learning and its performance measures, also going over the causes for poor performance, overfitting, and underfitting. Then deep learning, a subset of machine learning used usually for object detection is introduced, as well as a popular multilayer neural network, the convolutional neural network together with its architecture and layers. After this, a popular convolutional neural network ResidualNet is presented, as it was used as a feature extractor for this thesis. After a short excursion into computer vision with image classification, object localization, and object detection, the model used for this thesis, Faster R-CNN is looked at closer as well as TensorFlow and the TensorFlow object detection API, that was used for training.

In chapter 3 different approaches for localization, using different methods, are presented, including the approach the object detector developed in this thesis should be used for. As well as three different approaches for object detection itself with their advantages and disadvantages.

Chapter 4 deals with the acquisition of the dataset and the analysis of it. For this, a closer look at the classes and their distribution is taken. After this model selection as well as the model used in this thesis is discussed with its implementation and the training of the network.

And in chapter 5 the created object detector is evaluated using different metrics, from the PASCAL VOC and COCO datasets, designed for the comparison of different models. Also, a test video of a campus drive taken on another time than the videos for the dataset is used to investigate, how well the model does on unseen data. And at last, it is explored, if the model can be used to aid the localization algorithm proposed in [Kunz 21] to help the robot navigate.

At last, a summary of the thesis is given in chapter 6 as well as an outlook on future work.

2. Fundamentals

To understand the concepts applied in this thesis a well-founded understanding of some foundations is required. This chapter is an illustration of the most important ones, with many of them referenced later on. First basic concepts like Machine Learning and Deep Learning are explained. Then Convolutional Neural Networks and specifically ResNet are examined in more detail. After this Computer Vision and Object Detection are looked at as well as Faster R-CNN. At last, the system that was used to train the model, TensorFlow, and the Tensorflow Object Detection API are presented.

2.1 Machine Learning

With the help of Machine Learning, a lot of problems can be solved, that are natural to humans but hard for computers. Many are in the field of vision, pattern recognition, speech recognition, and robotics. It is a part of Artificial Intelligence and is currently enjoying great popularity. A definition from 1997 by Tom Mitchell gives a rough idea, what machine learning entails [Mitchell 97]:

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Task T

The above-referenced task t means not the process of learning but describes the task the system should be able to perform in the end like object detection [Goodfellow 16]. These tasks could be done with machine learning or with just writing a program that does it manually, this depends on the kind of task and also on the difficulty of the task and the conditions of the used hardware. A machine learning task is often a description of how an example should be handled by the system [Goodfellow 16]. These so-called examples are a collection of features that should be processed by the machine learning systems, typically depicted by a vector $x \in \mathbb{R}^n$ with each entry of the vector being another feature [Goodfellow 16].

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Figure 2.1: Formula for Accuracy [Accuracy 20]

Performance Measure P

The evaluation of a machine learning system is an essential step to decide if the task can be performed with the system and how good the system is in performing the task. For each task, a specific quantitative measure of its performance must be designed [Goodfellow 16]. For classification or an object detection task like in this thesis, the performance is measured by measuring the accuracy of the model. Accuracy describes the proportion of examples for which a correct output has been produced by the machine learning system [Goodfellow 16] as can be seen in figure 2.1. But the accuracy measure is not enough for the entire evaluation of a network, since an uneven distribution of data on the classes of the dataset can give a wrong impression with the accuracy score. To prevent this two other measures, precision and recall are used to fully evaluate the performance of a model.

Precision and Recall are frequently used and interesting performance measures for object detection. To understand these some building blocks of the metrics are described here, true/false positives and true/false negatives. Machine learning systems make predictions of classes of objects for example. If the positive class is correctly predicted, which means, that a recognized object is there, it is called a true positive, similarly to if the model predicts the negative class, which means, that there is no recognized object, it is called a true negative [True/False Positive/Negative 20]. A false positive is, therefore, an incorrect prediction of the positive class, which means, that a recognized object is not there and a false negative is the incorrect prediction of the negative class, which means, that a not recognized object is there [True/False Positive/Negative 20]. To decide, whether a previously made detection is a true or false positive the confidence score and the Intersection-over-Union (IoU) score are needed. The confidence score denotes the probability of an anchor box containing an object. This is usually provided by the classifier part of the network. The IoU score looks at the predicted box and the grounding box. It gives the score of the intersection of the area of these two boxes divided by the area of the union of these. With that information, the definition of the measure precision as shown in figure 2.2(a) and the definition of the measure recall in figure 2.2(b) is easy to explain. For precision of a class, the number of true positives is divided by the number of true positives and false positives added together, which means that a score of 1.0 would state, that every object that was detected, was also relevant for the task. On the other hand for Recall of a class, the number of true positives is divided by the number of true positives and false negatives added together, which means that a score of 1.0 would state, that every relevant object for the task was detected.

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

(a) Formula for Precision [Precision 20] (b) Formula for Recall [Precision 20]

Figure 2.2

Experience E

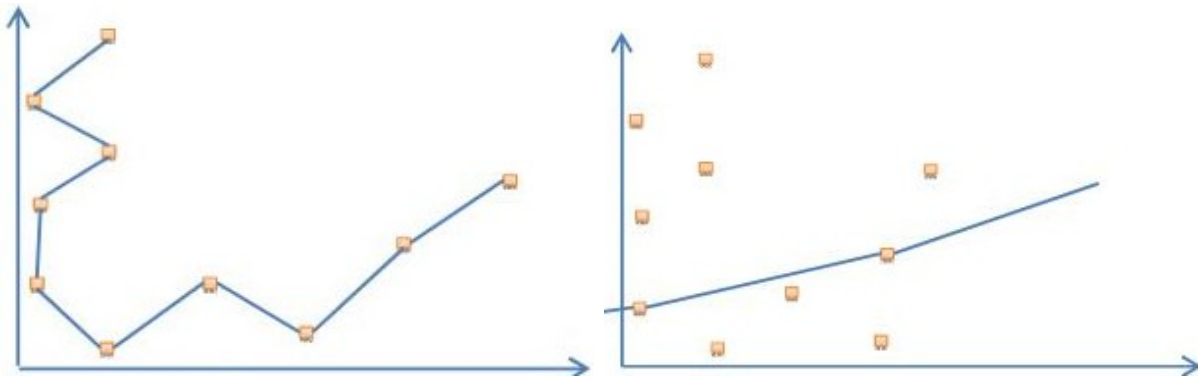
The Experience a machine learning system is permitted to have, decides about the classification of the learning as supervised, unsupervised, or reinforced [Goodfellow 16]. While classification, regression, and object detection, like many other well-known machine learning algorithms, are supervised learning problems, there are unsupervised learning problems like clustering and reinforcement learning problems like game playing [Alpaydin 14]. Since the model used and trained in this thesis experiences supervised learning, this technique will be looked at in more detail. Generally, the difference between supervised and unsupervised learning is the supervisor, which gives correct values to learn a mapping from input to output [Alpaydin 14]. This supervisor does not exist in unsupervised learning, where the aim is to find regularities in the input and so only input data is given [Alpaydin 14]. Reinforcement learning is different in the kind of output the system gives, since its more about a sequence of actions than a single move, hence game playing or a robot searching for a goal are good examples because the sequence of moves is more important than a single one and there is interaction with the environment[Alpaydin 14]. Supervised learning also has a different dataset than both previously mentioned methods. In the example are not only features but also associated labels with information [Goodfellow 16]. The task is to learn the mapping from the input x to the output y [Alpaydin 14].

2.1.1 Generalization

To use a trained machine learning system in a setting, that is new and has not previously seen inputs is an important challenge in machine learning. Generalization is the measure of how well a system can predict the right output for new instances [Alpaydin 14]. It is requested that the system can generalize well and still give a good performance.

2.1.2 Overfitting and Underfitting

Two causes for poor performance and generalization in supervised learning are overfitting and underfitting. If a model is flexible and without label noise, the training loss can be driven to zero, by remembering the correct output for each input [Murphy 22]. But as said in 2.1.1 the challenge is to predict accuracy on new data which is not part of the training data and is different from it. If the model fits the training set too well, because of noise like labeling errors and general memorization a phenomenon named overfitting can be observed [Jabbar 15]. Overfitting happens mostly when the dataset is too small, but other reasons



(a) Graph showing overfitting [Jabbar 15] (b) Graph showing underfitting [Jabbar 15]

Figure 2.3

can also come into consideration. When the model is trained for some time, there comes a point, where the ability to solve problems does not improve anymore but instead the model learns random uniformity contained in the training pattern [Jabbar 15]. This can be seen in figure 2.3(a) as the error on the test has a minimum, where the generalization ability of the network is at its best before it increases again [Jabbar 15]. So there is a gap between training error and test error, that is getting larger, not smaller. Underfitting is the opposite of overfitting and occurs if the error value on the training set does not get low enough [Goodfellow 16]. This phenomenon happens, when the variability of the data cannot be captured by the model [Jabbar 15]. Figure 2.3(b) shows, that the model is too simple to properly map and describe the training data and it will most likely have no predictive power [Jabbar 15].

After the introduction of machine learning as a rather general concept, a more modern and often used concept called deep learning is discussed in the next section.

2.2 Deep Learning

Deep Learning is a subset of machine learning, that is especially interesting if much data is available. This is because, as can be seen in figure 2.4, the accuracy depends on the amount of data, and the greater the amount of it, the better deep learning works as an improvement of machine learning. The models in deep learning are composed of multiple processing layers and therefore are called multilayer neural networks.

2.2.1 Multilayer Neural Networks

Multilayer neural networks consist of three types of layers, input layers, output layers, and between them are the hidden layers. From the input layer on the layers feed into the respective next one and all nodes in one layer are connected to those of the next layer [Aggarwal 18]. The input layer receives the input, performs computations, that are visible to the user, and then sends the data to the hidden layers, where an activation function produces an output from weighted inputs [Aggarwal 18]. The layers are called hidden

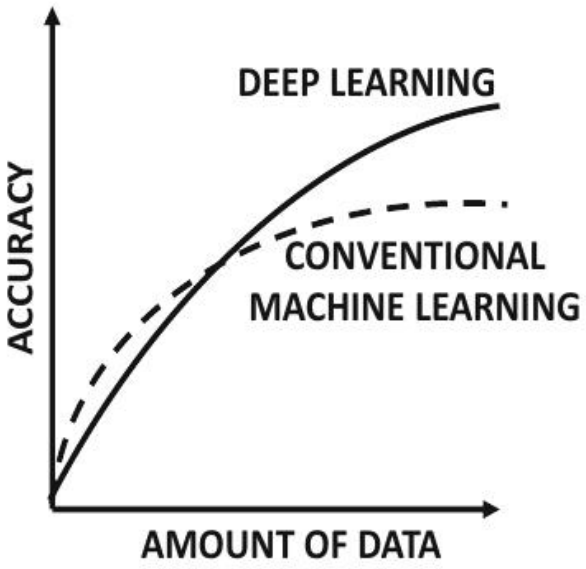


Figure 2.4: Comparison of accuracy in machine learning and deep learning[Aggarwal 18]

because the process is not visible to the user. The output layer only optimizes the loss function.

To go more in depth about the multilayer neural networks a example of these, convolutional neural networks are presented in more detail.

2.3 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a popular multilayer neural network that can process larger models with great amounts of data, which was not possible with the predecessor Artificial Neural Networks (ANN) [Albawi 17]. It is made to work on grid-structured data like images with a certain level of translation invariance, which means that the interpretation stays the same even if the location of the object in the image changes [Aggarwal 18]. To understand how CNNs are working first the overall architecture and the layers must be explained as well as the convolution, which gave the CNN its name.

2.3.1 Architecture

CNN's are composed of different kinds of layers, the convolutional layers, pooling layers, ReLU layers, and fully-connected layers, which when stacked form the architecture [O'Shea 15]. These layers have states, that are arranged in a 3-dimensional grid-structure [Aggarwal 18]. In figure 2.5 this structure is shown as having the three dimensions height, width, and depth, for images these values could be for example a width and height of 64x64 pixels, and a depth of 3 representing the RGB channel. So the depth refers to the number of channels in each layer or the number of feature maps in hidden layers and not to the depth of the network itself [Aggarwal 18]. The input layer decides these values when

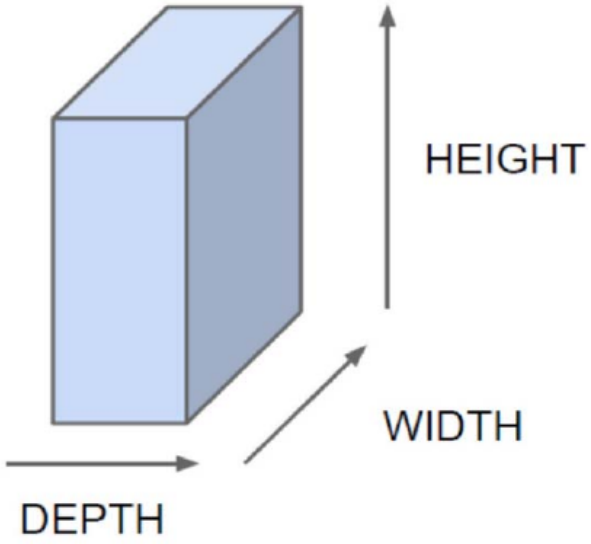


Figure 2.5: Example a 3-dimensional grid-structure [Albawi 17]

it is given the data. Parameters are sorted into filters or kernels, which are 3-dimensional structural units that are usually square and the depth is the same as that of the layer [Aggarwal 18].

2.3.2 Convolution

The convolutional layer usually comes directly after the input is given to the network. Defining for Convolutional Neural Networks is the convolution operation, which is useful for image data since it is a dot-product operation between a grid-structured set of weights and grid-structured inputs, which are derived from the spatial localities in the input [Aggarwal 18]. This operation can be used with different filters after one another and achieve a greater depth, with each output obtained from a single filter referred to as feature map [Aggarwal 18]. Convolution is used in place of matrix multiplication in the layers and mostly helps in denoising the input with the help of weighted functions [Goodfellow 16]. The filters or kernels are often called tensors. Convolution is important because it provides sparse interactions, parameter sharing, and equivariant representation and helps to make the model better [Goodfellow 16].

Sparse Interactions

In contrast to ANNs Convolutional Neural Networks don't use matrix manipulation to describe the interaction between input and output units but sparse interactions [Goodfellow 16]. This results in the usage of fewer parameters because not every pixel of a picture must be considered. This effect, evoked by using a filter that's smaller than the input, not only reduces memory but also has an impact on the statistical efficiency [Goodfellow 16]. To show these effects n inputs and m outputs are assumed. An approach using matrix manipulation would need $n * m$ parameters and the algorithm would have a

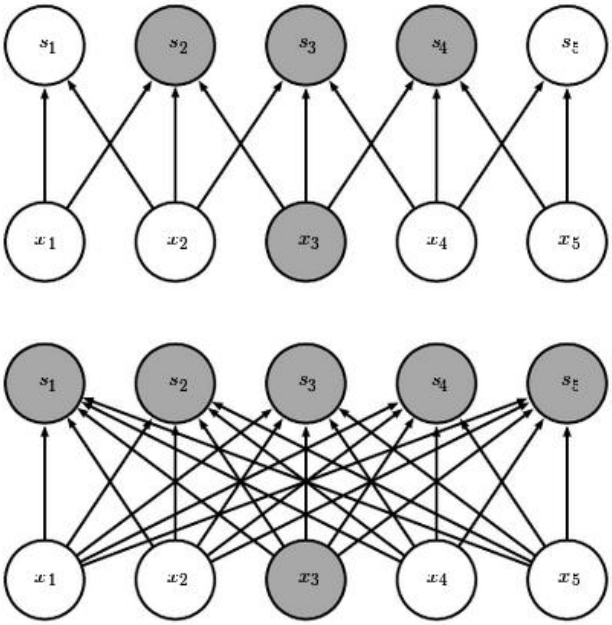


Figure 2.6: Comparison of Sparse Interaction with a kernel width of 3 (top) and Matrix Manipulation (bottom) [Goodfellow 16]

runtime of $O(n * m)$ per example. But an approach using sparse interactions and limiting the connections of each output to i, only $n * i$ parameters would be required and a runtime of $O(n * i)$ could be achieved [Goodfellow 16]. In figure 2.6 the input x_3 in the sparse interaction is shown to only affect three output units s_2, s_3 and s_4, whereas the same input in matrix manipulation affects all outputs.

Parameter Sharing

Another difference between ANN's and CNN's is parameter sharing. Normally parameters are not used for more than one function in the model, but in CNNs each member of the filter is used at every position in the input, which means that the model learns from one set rather than a set of parameters for every location [Goodfellow 16]. This has mostly an effect on the memory usage not on the efficiency. Since the runtime is still $O(n * i)$ but the storage is reduced to i parameters [Goodfellow 16].

Equivariant Representations

A special form of parameter sharing, called equivariant representation or equivariance to translation is used in the convolution and helps with dealing with change at the input. So if a pixel value in the input is shifted in any direction and then convolution is applied, the feature values will shift with the output values [Aggarwal 18]. This is useful because if the object is moved in the input, its representation will use the same amount in the output [Goodfellow 16].

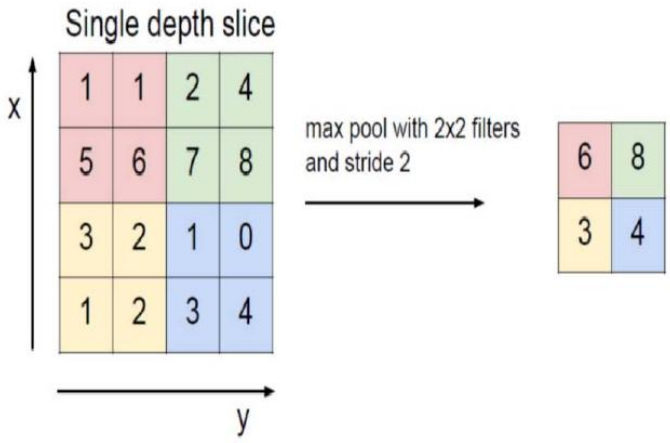


Figure 2.7: Max-pooling with 2x2 filter and stride 2 [Albawi 17]

There are more optimizations for the complexity of the model that take place in the convolutional layer like tuning the parameters depth, stride, and zero-padding. While reducing the depth of the output produced by the convolution can minimize the number of neurons in the model, it can also reduce the pattern recognition capabilities of the model and should be handled with care [O'Shea 15]. Defining the stride to set the depth around the dimensions of the input is easier to use and a greater stride will produce a lower spatial dimension output by reducing the amount of overlapping [O'Shea 15]. Zero-Padding is also used to change the dimensions of the output volumes by padding the border of the input [O'Shea 15].

2.3.3 Pooling

The pooling operation is nested in the convolution operation and produces a layer with the same depth [Aggarwal 18]. But it is used to further reduce the computational complexity of the model. This is done by scaling down the activation map [O'Shea 15]. In an approach called max-pooling shown in figure 2.7 the maximum output within a rectangular neighborhood is reported for each activation map [Goodfellow 16]. This is needed because if the input is translated by a small amount the value of the pooled outputs will not change [Goodfellow 16].

2.3.4 Rectified Linear Unit

As the pooling operation, the Rectified Linear Unit (ReLU) operation is also nested in the convolution operation and is not very different from the ReLU activation in ANNs [Aggarwal 18]. There is a one-to-one mapping of activation values and thus does not change the dimensions of a layer [Aggarwal 18].

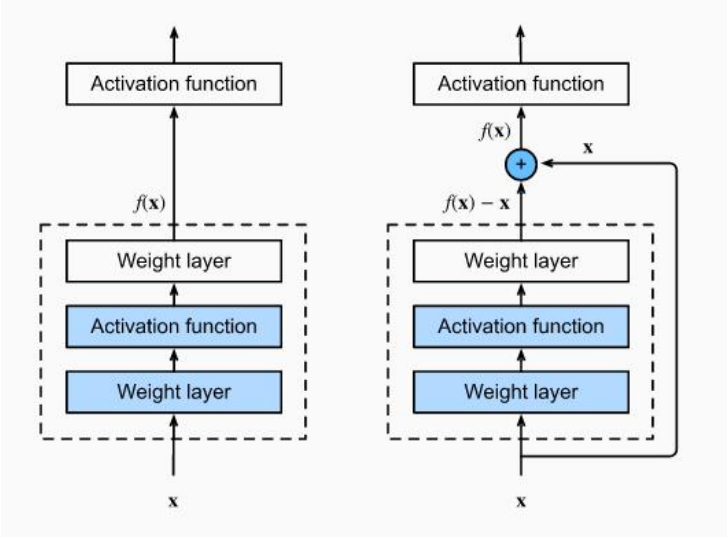


Figure 2.8: A regular block (left) and a residual block (right) [Zhang 21]

2.3.5 Fully Connected Layer

Fully Connected Layers are usually connected to each feature in the final spatial layer and function like feed-forward networks [Aggarwal 18]. Mostly more than one fully connected layer is used and so the power of computations can be increased towards the end [Aggarwal 18]. Fully connected layers are specifically designed for each application and the activation methods, like logistic or softmax, are also chosen depending on the application [Aggarwal 18]. These layers also have the disadvantage of being very complex and including a lot of parameters and in 2.3.6 it is discussed how ResNet deals with this problem.

2.3.6 ResNet

ResidualNet (ResNet) is a model by a team at Microsoft, which won the 2015 ILSVRC and COCO competitions in the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation [He 16].

Residual Learning

In the design process of deep neural networks, a challenge arises in the form of adding more layers. This leads to a degradation problem, where the accuracy gets saturated but then degrades and the training error is higher [He 16]. The solution to this problem is to construct added layers as identity mappings to guarantee a training error not greater than that of a shallower model [He 16]. So residual learning is used every few stacked layers to skip connections between layers and introduce an iterative view of feature engineering [Aggarwal 18]. These shortcuts between layers are not increasing complexity or create extra parameters. In figure 2.8 a representation of residual learning is given. Assume the input to be x and the mapping that should be obtained by learning to give to the activation function to be $f(x)$. In the right block, the advantage of residual mapping becomes clear,

because there is a shortcut function to forward through the residual connections across layers [Zhang 21]. These shortcuts can be used if the input and output are the same dimensions [He 16].

Architecture

This residual network lies on top of a plain baseline, that orientates itself at VGG nets with a convolutional layer of 3x3 filters with the same number of outputs, and downsampling is performed by the stride of 2 [He 16]. Then comes a global average pooling layer and a 1000-way fully connected layer with softmax [He 16]. The architecture can include 18, 34, 50, 101, or 152 weighted layers. In this thesis, a ResNet-50 network was used, because it is faster than the network with 101 or 152 layers but still provides high accuracy and the architecture of this network deserves a closer look. The blocks in ResNet-50 are 3-layer bottleneck blocks, which consist of the convolution layers 1x1, 2x3 and 1x1 with the first and last layer responsible for reducing and restoring the dimensions and leaving the middle layer a bottleneck with lower input and output dimensions [He 16].

After this presentation, now a short detour to computer vision will be taken. This is necessary to understand the application part of the work, which is about detecting objects in images and videos.

2.4 Computer Vision

Computer Vision is a part of Computer Science and has many applications. Mostly in object detection or object localisation, which are elaborated in 2.4.2 and 2.4.3, but also for example in medical imaging, mobile robot navigation and human-computer interaction. But the field is also interdisciplinary having overlaps with geometry, physics, and learning theory [Forsyth 11]. The challenges of computer vision are often closely connected to these fields. Some common difficulties include illumination, different viewpoints and intraclass variations [Zou 19]. Some applications of computer vision, that are used in the thesis are discussed in the next sections.

2.4.1 Image Classification

In image classification, a whole image is shown and then a single label for this image should be associated with it. A well-known example is the cat vs. dog classification, which labels images with "cat" or "dog" if one can be seen in it. But this classifier can only label the image with one class and what first seems to be an easy challenge proves to be quite difficult, if more than one object can be identified in the image. This problem is known as image tagging and the output space is now defined as $y = \{0, 1\}^C$ where C is the number of tag types [Murphy 22]. So one image can have multiple tags if more than one object can be seen in the image. This means that the model doesn't have to decide which label to use because the option to use more than one is available.

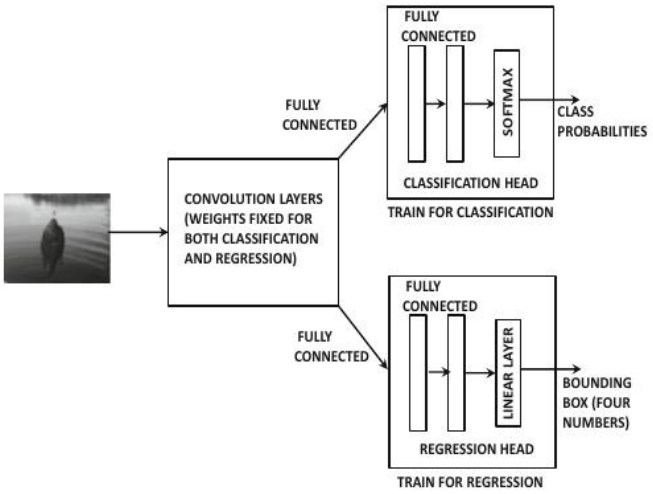


Figure 2.9: Broad depiction of the layers of a object localisation model [Aggarwal 18]

2.4.2 Object Localization

Object Localization is part of the field of Computer Vision and expands the image classification since the question is not only what object can be seen in the image but also where it is. Localization works only on a fixed number of objects in an image and usually first classifies the object and then draws a bounding box around it [Aggarwal 18]. Since the bounding box, which can be identified using four unique numbers, is a regression problem only one model is needed [Aggarwal 18]. This is because of the generalized convolutional neural network, which can do both, classification and regression. It only varies in the final two fully connected layers, where a split is done as can be seen in figure 2.9.

2.4.3 Object Detection

As described above, object localization and object detection are closely related, but detection can be described as the more difficult problem of them both. The number of objects in the image that should be classified and localized is no longer a fixed number but a variable number of objects of different classes [Aggarwal 18]. This variable number of objects can be solved for example by using a sliding window approach or region proposal methods. In the sliding window approach, all possible locations and scales in an image are visited, which means that every bounding box that is possible is tried and, then the localization is used to detect objects [Aggarwal 18]. This approach has a few problems, which can impact the response time of the model. For example, an image of size $n*n$ would have a possible number of bounding boxes of n^4 and, for all of these, a classification has to be performed [Aggarwal 18]. This problem is solved with region proposal methods, which creates larger regions of similar pixels and thus has a smaller set of proposed bounding boxes [Aggarwal 18].

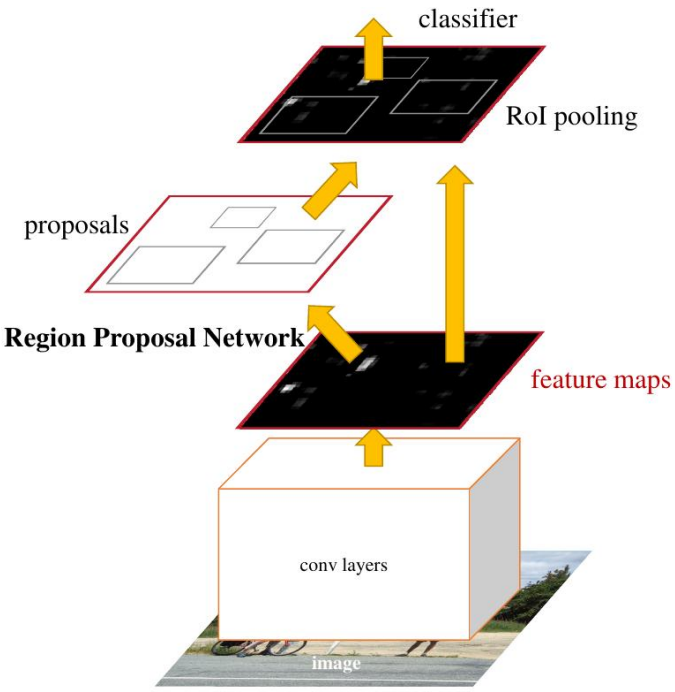


Figure 2.10: Faster R-CNN with its two modules for object detection [Ren 15]

2.5 Faster R-CNN

Faster R-CNN depends on region proposal networks to predict the location of objects and a Fast R-CNN detector that uses the proposed regions [Ren 15]. This structure can be seen in figure 2.10, where the models of region proposal and classification are combined into one network for object detection.

2.5.1 Fast R-CNN

Fast R-CNN is the improvement of R-CNNs, which had some problems like slow object detection expensive training and multi-stage pipeline training [Girshick 15]. The architecture of Fast R-CNN is shown in figure 2.11 and now described further. The input of this network is an image and a set of object proposals, of which first the image is processed with convolutional and pooling layers to create a feature map [Girshick 15]. From this feature map, the region of interest (RoI) pooling layer extracts for each object proposal a fixed-length feature vector to feed into fully connected layers [Ren 15]. In the figure 2.11 it is shown, that there are two output layers. One gives a softmax probability vector and the other a per-class bounding-box regression offset vector [Girshick 15].

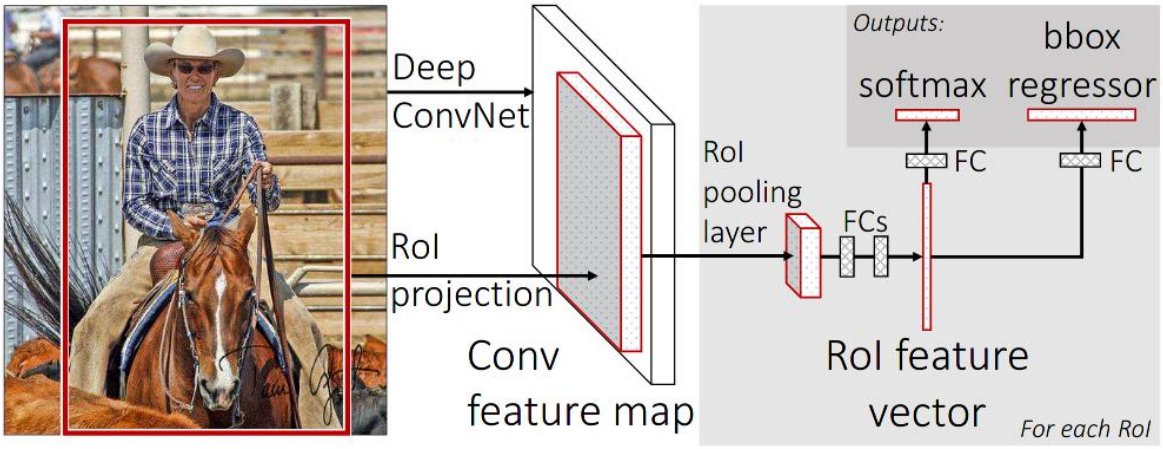


Figure 2.11: Fast R-CNN architecture [Girshick 15]

Region of Interest

As the name says, a region of interest is not the whole feature map, which is acquired from the convolutional layers, but just a region of it, which seems to be important or of interest. These regions can help reduce the feature maps to smaller ones, which happens in the RoI pooling layer. Since each RoI is characterized by a tuple (r, c, h, w), with h = height and w = width, the layer divides the region into smaller regions and then max-pools the values in each sub-region into the respective output [Girshick 15]. This improves mainly the speed of detection since there is a preselection of regions to consider and not the whole image.

2.5.2 Region Proposal Networks

A region proposal network (RPN) is modeled by a fully convolutional network that takes an image as input and gives a set of rectangular object proposals as output [Ren 15]. To share the computation with the Fast R-CNN network they are assumed to have a common set of convolutional layers [Ren 15]. A sliding window approach is used on the convolutional feature map output of the last convolutional layer that is shared and takes an $n * n$ spatial window of the input convolutional feature map [Ren 15].

Anchors

An anchor is a reference box in which the proposals from the RPN are parameterized relative to [Ren 15]. For each sliding window, an anchor is centered in it and multiple region proposals are predicted [Ren 15]. So if three scales and three aspect radios are used, which are associated with the anchor, nine anchors would be given at each sliding window position [Ren 15]. The Anchors in Faster R-CNN are also translation invariant, which means, that if an object in an image is shifted, the proposal should be predicted in either position [Ren 15].

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

Figure 2.12: Loss function for an image in Faster R-CNN [Ren 15]

Loss Function

In figure 2.12 the loss function used in Faster R-CNN can be seen. To understand this formula at first some definitions must be made and then the elements must be explained. The definitions concern the objective function, which should be minimized, meaning a search for the set of weights with the lowest score. If the objective function is minimized it is called the loss function and the value calculated by it is the loss, which represents the difference between prediction and ground-truth. A binary class label is assigned to each anchor, when training an RPN, to show that it is an object or not [Ren 15]. In [Ren 15] a positive label is given to two kinds of anchors:

1. the anchor/anchors with the highest Intersection-over-Union (IoU) overlap with a ground-truth box or
2. an anchor that has an IoU overlap higher than 0.7 with any ground-truth box.

In the formula 2.12 p_i is the predicted probability of am anchor i being an object or not, with the ground-truth label $p_i{}^*$ being 1 if the anchor is an object and 0 if this is not the case [Ren 15]. Similarly t_i is a representation of the predicted bounding box, with $t_i{}^*$ representing the ground-truth box vector associated with the anchor [Ren 15]. L_{cls} represents the classification loss and L_{reg} the regression loss, with the regression loss only activated for positive anchors [Ren 15].

2.5.3 Sharing Features

After Fast R-CNN and RPNs are explained there is only left to show how they are brought together to form Faster R-CNN. These two networks have to be trained together otherwise a formation of different convolution layers could not be prevented. One technique used in the original paper is the 4-step alternating training. There first the RPN is trained, then the Fast R-CNN is trained with the proposals generated by training the RPN, then the detector network is used to initialize the RPN training again, but only the layers belonging to the RPN are fine-tuned and at last only, the layers of the detector network are fine-tuned [He 16]. This leads to a common convolution layer and in the end, a combination network is formed out of the two networks.

To train a network like Faster-RCNN the help of TensorFlow can be utilized, which is now presented in more detail.

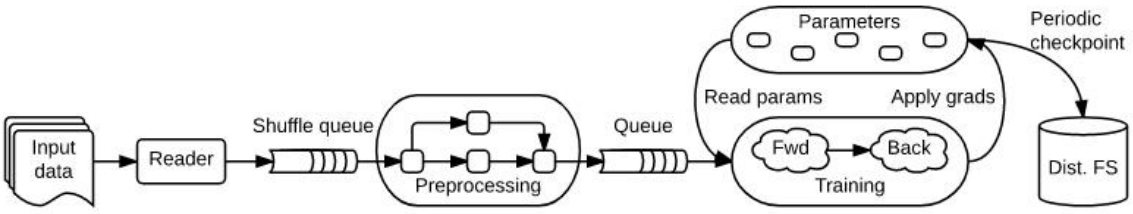


Figure 2.13: Schematic TensorFlow dataflow graph for a training pipeline [Abadi 16]

2.6 TensorFlow

TensorFlow is a core open-source library to train and develop different machine learning models. It operates at a large scale and uses dataflow graphs to represent the computation [Abadi 16]. It is designed to be flexible and allows users to deploy applications on different devices or workstations [Abadi 16]. The experimentation with different models works without changing the core system because of a high-level scripting interface, which wraps around the dataflow graph [Abadi 16]. A representation of a dataflow graph can be seen in figure 2.13. Within this graph all states in a machine learning algorithm are contained, like the input preprocessing, parameters and their update rules and all computations [Abadi 16]. Each vertex in this graph indicates a local computation and each edge indicates the output and input of a vertex [Abadi 16]. In 2.13 one can also see the reason for the flexibility of TensorFlow, the concurrent execution. This technique allows subgraphs to be executed at the same time and also makes interaction possible with the help of shared variables and queues [Abadi 16].

Tensors

Tensors are an elemental part of TensorFlow, as they model the data as n-dimensional arrays with the content having a primitive type such as integer, float, or string [Abadi 16]. These are needed to compute mathematical operations like batch convolution or matrix manipulation. They are also useful because TensorFlow can be used on CPU, GPU, or special TPU (Tensor Processing Unit) devices and tensors can be understood by all of them.

TensorFlow Object Detection API

For this thesis, the TensorFlow object detection application program interface (API) is used. This API is also open-source and build on top of TensorFlow to help train and deploy object detection models. It provides different meta-architectures, like Single Shot Multibox detector and faster R-CNN, to combine with different feature-extractors, like VGG and ResNet, as well as a collection of pre-trained models.

3. Related Work

This chapter gives first an overview of outdoor localization and then one of the state-of-art of object detection. Both topics are important parts in the application and research of autonomous driving but also other areas. While in object detection there are many possible solutions for the task, three models that assert themselves because of their performance, are now presented with their strengths and weaknesses. In outdoor localization, there are also many different possible approaches, that will be presented. Some of them are better suited to certain tasks than others and it is important to find the right one out of all the possibilities.

3.1 Outdoor Localization

An example for a localization method, that does not use the Global Positioning System (GPS) information but odometry measurements, compass, and sparse-map-based measurements updates can be found in [Chipka 18]. Here a pose estimator estimates the pose of the vehicle within a frame of reference using the start and goal locations, dead reckoning odometry, and compass measurements as well as a sparse map of landmarks with corresponding images and coordinates in its database. This system only works for shorter routes, depending on the landmark detection rate and landmark density but it also assumes a failure of GPS and a lack of a map. Because of this, it is not applicable as a localization method for the robot driving on the campus of a university.

In [Nilwong 19] a localization method is proposed, that relies on a landmark detector and a feed-forward neural network (FFNN) trained with GPS data from geotags in images, which then retrieves location coordinates and compass orientation based on the landmarks. This is an example of a visual-based localization that also uses GPS information, but only in the training of the FFNN. In figure 3.1 the structure of the model is shown with the two networks. The Faster R-CNN part gives bounding boxes, labels, and scores to the FFNN, which generates geolocation data. The model shows a good performance with small error, but the area in which it was tested was rather small and so some more errors in a bigger area could be expected. A problem with this model exists, which makes it

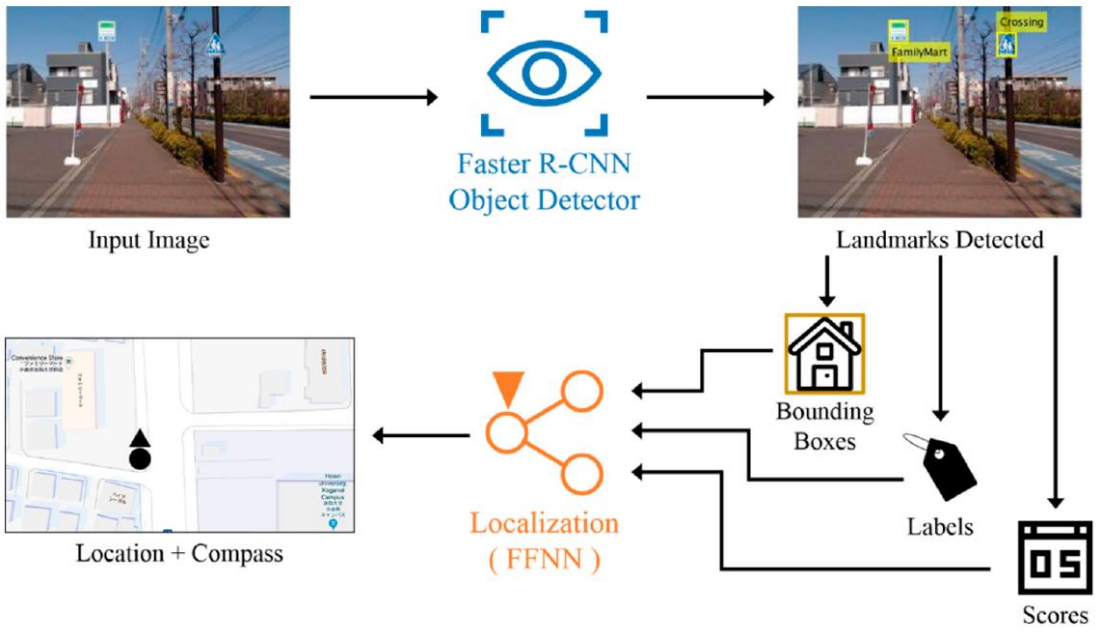


Figure 3.1: Faster R-CNN based localization method as proposed in [Nilwong 19]

hardly usable for the task of this thesis. On the side of a road, many landmarks can be detected, these include but are not limited to advertisements, signs, and lamp posts. On a university campus is not necessarily an abundance of such landmarks to be detected and in the worst case, the vehicle might see nothing but bushes and trees for some frames. This poses a problem for localization only depending on landmarks.

An approach presented in [Kunz 21] by Maximilian Kunz of the Robotics Research Lab of the Technical University of Kaiserslautern uses OpenStreetMap (OSM) data to supplement the localization given by the GNSS. OSM uses nodes to mark elements like artwork, building entrances, fire hydrants, and other landmarks. As is explained in [Kunz 21] there are at least two of these nodes needed for the localization algorithm, which returns an estimated position based on the matches of the detected landmarks and the ones in OSM. This approach does not only depend on landmark recognition or a map but a map, GNSS, and landmark detection. This makes it a very robust approach because there are many possibilities to update the location of the robot.

This thesis aims to extend the proposed method by an object detector to detect the OSM landmarks on the campus, that can be used with the existing algorithm and the OSM data. Therefore the detector has to provide the tag, confidence, bounding box and local position to identify and compare the objects [Kunz 21]. So there is a greater focus on the detection part, which will be looked at in greater detail than the localization.

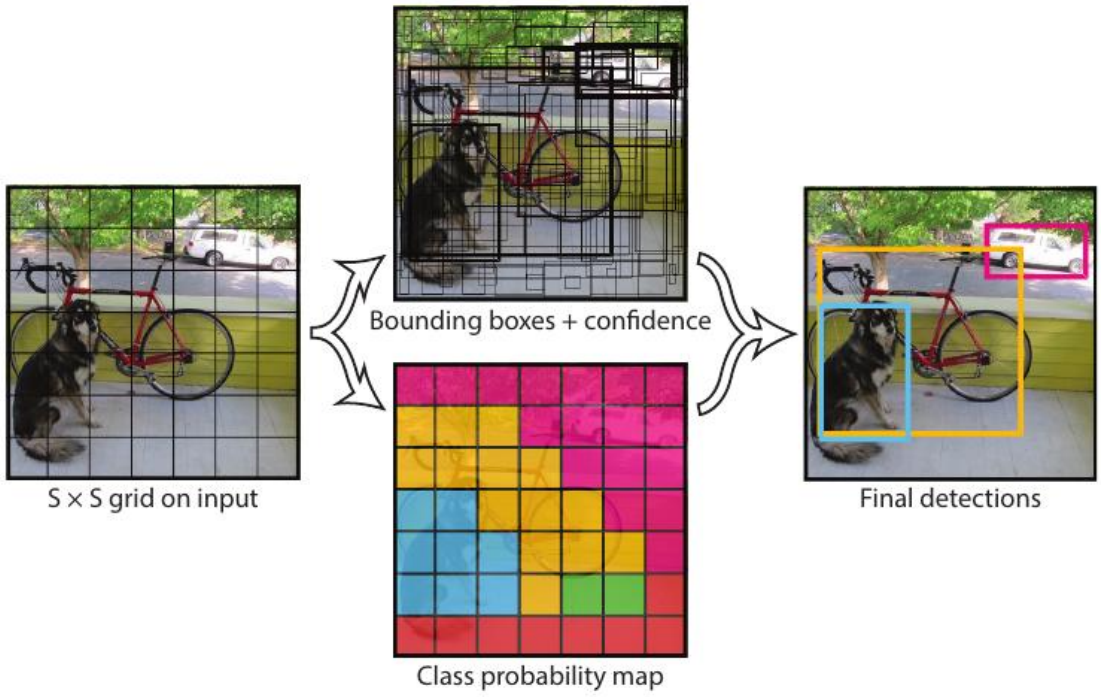


Figure 3.2: YOLO model object detection[Redmon 16]

3.2 Object Detection

Object Detection is a technology, that is a part of the field of Computer Vision. As the name is implying, the task of object detection is detecting instances of visual objects in digital images and videos [Zou 19]. Even though this does not seem too hard for a human, the task itself is rather hard for computers.

Today, most of the interest in object detection comes from a type of research topic, named "detection applications", containing specific application scenarios, for example pedestrian detection, traffic sign detection, and text detection, with another type of research the topic being "general object detection", simulating human vision and cognition [Zou 19]. As the examples show, object detection is a major part of deep learning and is utilized in fields like autonomous driving and localization. In the history of object detection many different approaches are used, from traditional detectors like Viola-Jones Detectors or the HOG Detector or the Deformable Part-based Model (DPM) to CNN based Two-stage Detectors and CNN-based One-stage Detectors, that have better performance [Zou 19]. There are two recent one-stage detectors, You Only Look Once and the Single Shot MultiBox Detector, that comes into consideration to use in this thesis and should be looked at in greater detail. As well as a two-stage detector, Faster R-CNN, which will be compared to the other two.

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Figure 3.3: Comparison on Pascal VOC2007 metric [Liu 16]

You Only Look Once

In 2016 a new approach to object detection was presented called You Only Look Once (YOLO) [Redmon 16] that addresses object detection as a single regression problem. YOLO uses only one convolutional network to predict bounding boxes and class probabilities, which makes it very fast. In figure 3.2 an example for detection can be seen. The image gets divided into an SxS grid and the network predicts bounding boxes and confidences for these as well as class probabilities. This proceeding comes with limitations because the grid structure limits the number of predictions per box and classes per box. This means that if many objects are close together, especially if they are small, the model struggles.

Single Shot MultiBox Detector

Also in 2016 the Single Shot MultiBox Detector (SSD) [Liu 16] was presented. It produces a fixed size of bounding boxes and scores for class instances in the boxes and gives final detections with the help of a non-maximum suppression step. The base network is based on the standard architecture for image classification, which is then supplemented with three key features, multi-scale feature maps, convolutional predictors for detection as well as default boxes and aspect ratios [Liu 16]. Multi-scale feature maps are used for detection at multiple scales by adding progressively decreasing convolutional layers. The convolutional predictors produce detection predictions on each feature layer using convolutional filters. And the default boxes fix the position of the boxes relative to the default box shape in the cell.

Faster R-CNN

Faster R-CNN [Ren 15] is an object detection network presented in 2015 that depends on region proposals algorithms. The computational cost of region-based CNNs was reduced by convolution sharing across proposals and in Faster R-CNN nearly cost-free region proposals are possible, which helps it in achieving near real-time rates [Ren 15]. A more detailed description can be found in 2.5.

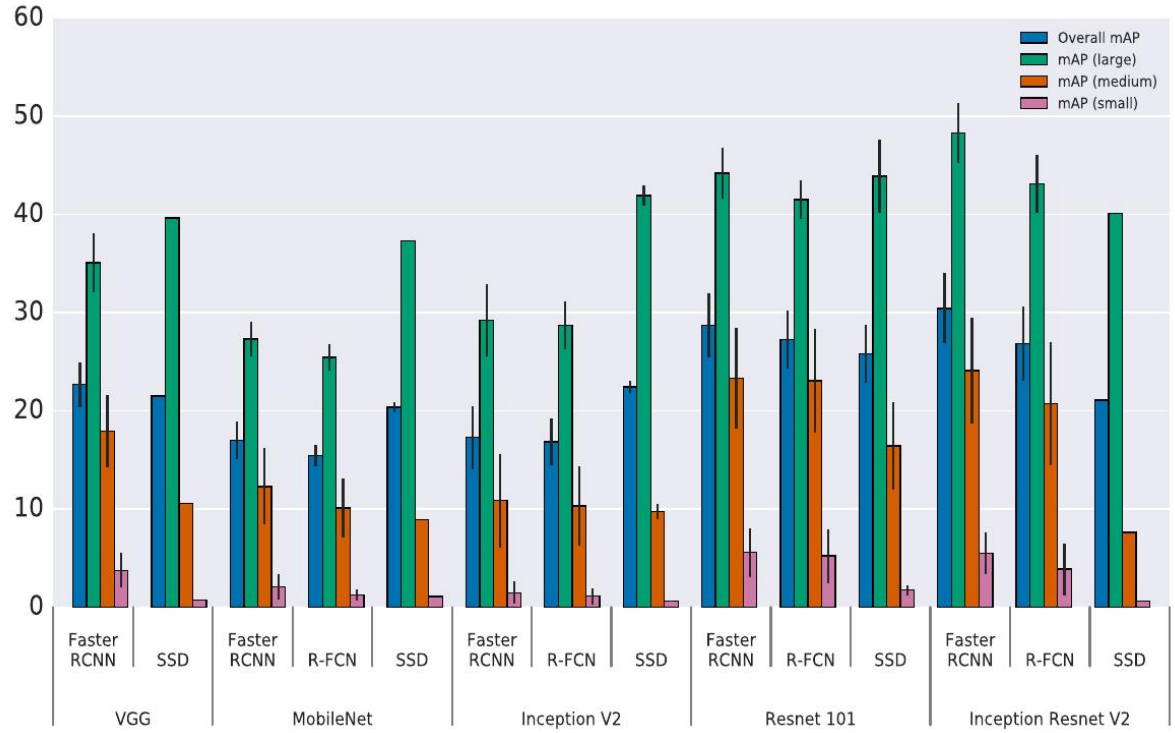


Figure 3.4: Comparison of accuracy stratified by object size, meta-architecture and feature extractor [Huang 17]

For this thesis, the Faster R-CNN network was used, because it has higher accuracy than the other two. While YOLO is very fast, its accuracy is in comparison too low for the task of landmark detection. Also, detection in real-time is not relevant for the model, because the localization is not made every 5ms but only when the scene is changing. As can be seen in figure 3.3 the mean average precision (mAP) of YOLO is worse than that of Faster R-CNN and SSD. There can also be seen, that SSD300 is even faster than YOLO while having an mAP that is comparable to that of slower techniques like Faster R-CNN. But in figure 3.4 a problem of SSD becomes visible.The mAP for the detection of small objects is noticeably lower with SSD than with Faster R-CNN. Since also smaller landmarks like waste bins, fire hydrants, and building numbers should be detected, which can be really small depending on the distance, Faster R-CNN seems to be the better choice.

4. Implementation

In this chapter, the way from the preparations to the finished model is explained. To reach the goal of OSM landmark detection a few steps have to be taken. Starting with the dataset used to train the network as well as a description of how it was acquired and processed. Then an explanation for the selection of the used model and the way it works and is implemented is given. At last information about the training process is given.

4.1 Dataset

Data is very important in deep learning and with the rise of more application areas, there was also a rise in usable datasets. Object detection has seen the rise of many big datasets, which are used to train models and also to evaluate them. With the help of these datasets, computer vision and its subsets have become a field that attracts many researchers. Big datasets are needed to train models with high accuracy and generate pre-trained models, which can be trained further. One of these datasets, the COCO dataset, is now described closer since it also plays a part in training the model for this thesis. Though it doesn't concern the finished network, the so-called transfer-learning utilizing a pre-trained model helps with the training of the network, which has a smaller dataset.

Microsoft COCO: Common Objects in Context

The COCO dataset is designed for the detection of objects in their natural context [Lin 14]. It has more instances per category than other datasets used for training and evaluation, like PASCAL visual object classes (VOC), as can be seen in figure 4.1. This is helpful for precise localization and also helps with the generalization capabilities of the model [Lin 14]. Another distinction from other datasets that helps with generalization is the focus on the detection of non-iconic views of objects and the precise 2D localization of objects [Lin 14]. Non-iconic views are not restricted to unobstructed objects in the center of the image, but also include partial objects, objects in the background, and cluttered scenes [Lin 14]. This is a more real-life, everyday-orientated approach.

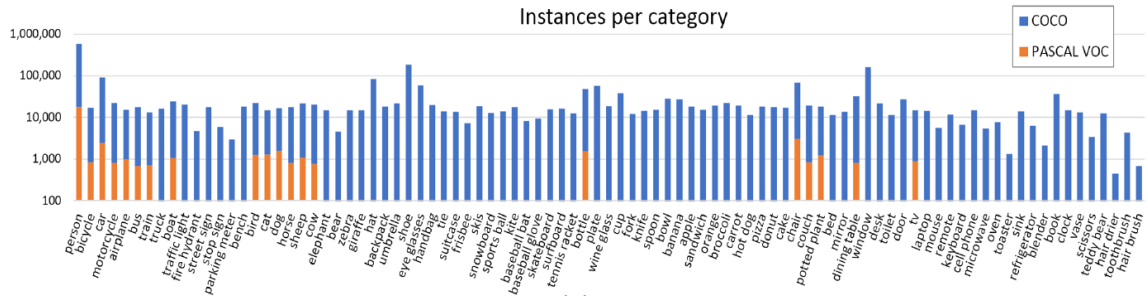


Figure 4.1: Comparison of instances per category in COCO and PASCAL VOC [Lin 14]

Because of this, the COCO dataset is chosen to be the dataset the pre-trained model is trained on, which should then be used for training the model in this thesis. With the focus on the 2D localization and non-iconic views of objects, it emphasizes things, that are also important for the model in this thesis, so the use of transfer learning seems good to use in this context. Transfer learning is a recognized technique to utilize a model, that is already trained for a similar problem. Especially for object detection and image recognition, the use of transfer learning is great, since the model then needs less training and usually yields better results, as well as utilizing an already recognized architecture, that is proven to work. The model gets initialized with pre-trained weights and then trains further with the self-created dataset, which is described now in greater detail.

4.1.1 Data Acquisition

In preparation for the creation of the dataset used for this thesis, possible classes are considered. These classes should orientate themselves on the landmarks in OSM as well as other well recognizable objects. Landmarks in OSM are visible on the map 4.2 and include building entrances, waste bins, hydrants, artwork, benches, fitness stations, and many more. But for this thesis, only the named ones are considered. Also considered are recognizable objects like the types of buildings, since they differ from one another. This means that buildings with numbers in the 40s look alike as well as buildings with their numbers in the 30s and the 10s. A visualization of which buildings belong to which type can be seen in figure 4.3. Also, some buildings differ from each other and the ones with the same number range, so they get their own classes, like the central university library, the bistro36, the mensa, and the buildings 42 and 47. Also, the numbers of the buildings can often be seen and be used for localization. In figure 4.12 a listing of all classes recognizable with this detector can be seen.

The dataset is then created by driving the route marked in figure 4.4 on the campus of the TU Kaiserslautern. The robot used for this task is the Delivery-Robot shown in figure 4.5(a), which was designed to transport heavy loads on the campus. For this dataset only the possible routes on the outside of buildings are considered since within the buildings are no OSM landmarks that are considered for the localization of the robot at this time. Also, the robot has to be able to drive the possible routes, which means that no points only reachable by stairs are on the route. Since the robot itself is used to generate the image data all of the buildings, which can be reached by the robot on the selected route are

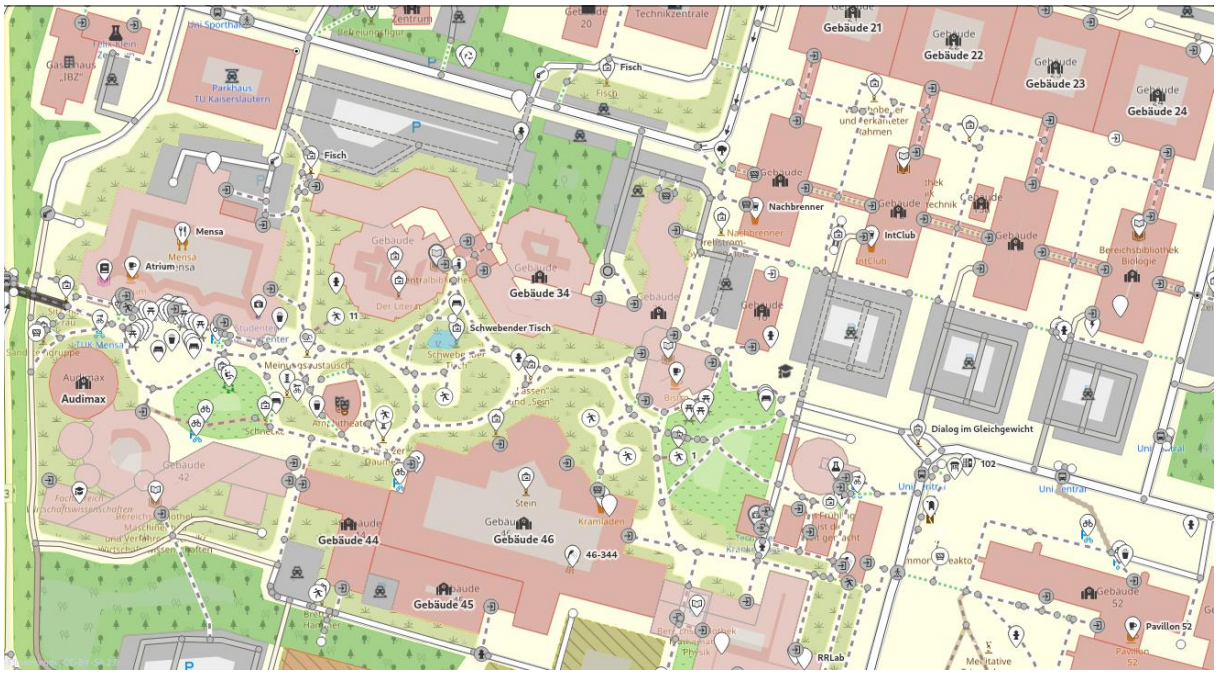


Figure 4.2: Marked objects in OpenStreetMap

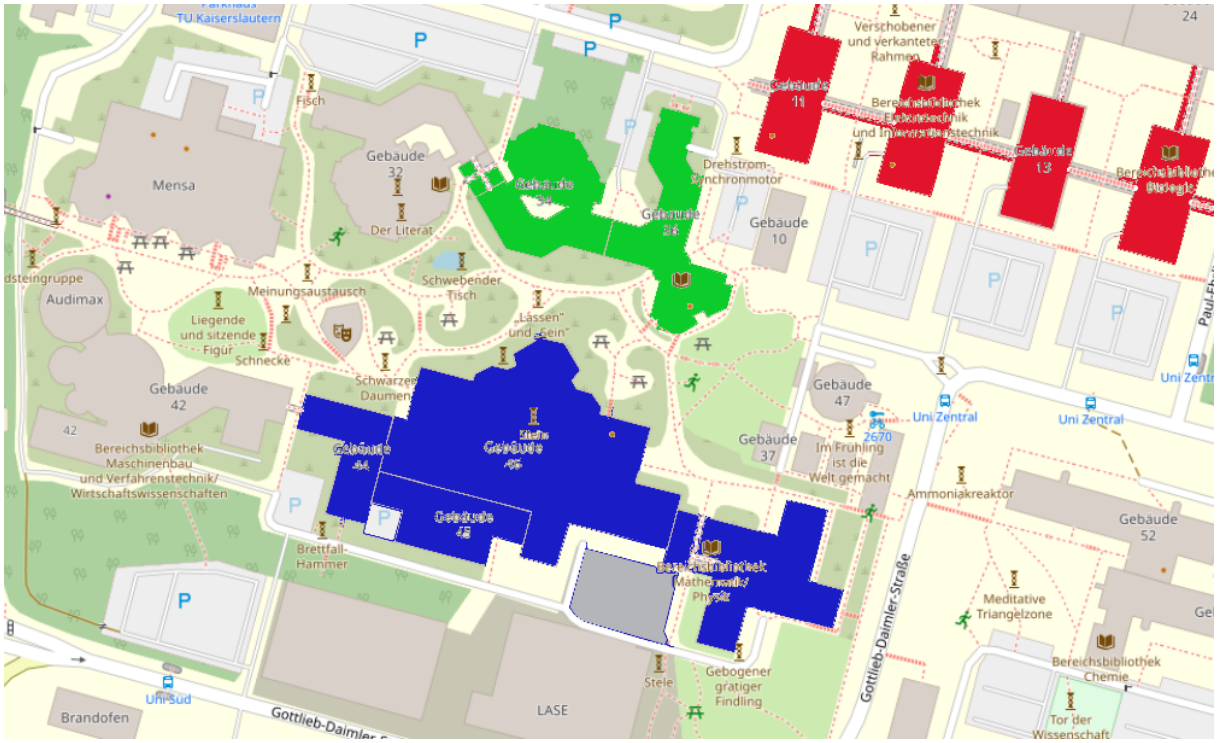


Figure 4.3: Map that shows the different building types. Buildings of type 30 in green, buildings of type 40 in blue and buildings of type 10 in red.

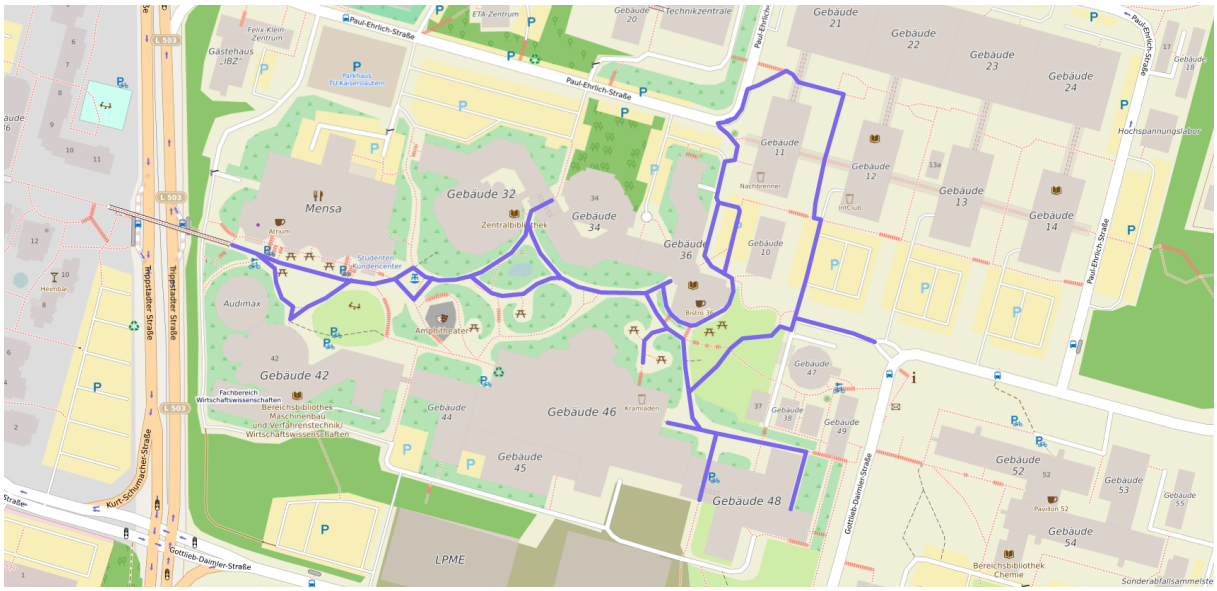


Figure 4.4: Route that was driven on the campus to acquire the images for the dataset

covered. The two selected routes are inspired by the most popular paths on the university site, which can be reached by the robot. One from the RRLAB entrance in Building 46 to the Mensa and Building 42 and the other from the RRLAB entrance to Building 11 which can be seen on the map in figure 4.4. On these routes, many landmarks, that are marked in OSM, can be seen and every class considered before is covered.

The robot that is used for the creation of the dataset has dimensions of 100cm x 70cm x 110cm, which makes it possible, to drive along smaller paths on the campus, normally used by pedestrians and cyclists. The robot is equipped with 2 Sick TiMS as safety sensors and an Ouster OS0-128 for monitoring of more distant objects, as well as for localization support [PostBot 21]. More important for the task of this thesis, the robot also has two ZED2 stereo cameras, which can be seen in figure 4.5(b). One is on the front of the robot and one on the back, which makes it easy to generate different views of objects in one drive. Also, there are no images of places, that the robot can't access, so theoretically the whole video is used to generate images for the dataset. After two drives, covering the route in figure 4.4 over 90.000 images were generated. These drives take place on a sunny day in June 2021 and a cloudy day in August 2021. To have more weather variations a third video of a drive in June 2020 is used, which has a mixture of the weather of the other two videos. Other weather situations are not considered at this time, because the robot can't drive in the rain and snow at this point. Of course, all these videos generate too many images to use and in the next section, the processing of this raw data to a usable dataset is explained further.

4.1.2 Data Preprocessing

The data for the training of the model is very important and has to be closely examined and controlled to generate a good working model. Therefore not all of the 90.000 collected images are suitable for the task. Because of the movement of the robot many images are

(a) The Delivery-Robot used to acquire the images for the dataset [PostBot 21]

(b) The camera used to acquire the images for the dataset [ZED 21]

Figure 4.5

too blurry to detect anything in them. This task can be hard even for humans and the annotation should be with as few errors as possible to train the network. Also, there are images where the objects are too occluded or only a small part of it can be seen. These are also not suitable, because the area is just too small to detect them without a doubt. Sometimes objects are so far away that they are not really visible and such images are also prone to give false detections later on. The dataset ready for annotation is achieved by considering every image and only keeping the ones that seem suitable. To summarize the selection process the rules for the images are

1. Every image with motion blur is discarded
2. Every image with an object smaller than easily visible to the naked eye is discarded
3. Every image with partial objects is discarded if only half or less of the object is visible

After sorting out all of these images as well as the ones, which are basically the same, because of the slow speed of the robot, the dataset still has 13.466 images that should be labeled and used for the training.

Annotating Images

LabelImg [Tzutalin], an image annotation tool is used to annotate images and save the annotations as XML files in PASCAL Visual Object Classes (VOC) format, which is needed to train the network with the TensorFlow API. For every image, an XML file is generated. In this file, the *ground_truth* of the dataset is saved to use in training and validation. This *ground_truth* consists of bounding boxes for each object visible in the image and labels, that associate the bounding box with the object in the bounding box, by defining its class. A bounding box is a rectangle defined by width, height, and the x and y values of the top-left point of the rectangle called xmin and ymin. as well as the x and y values of the bottom right point of the rectangle, called xmax and ymax.

In the next section, the dataset is looked at in more detail after the explanation of the image augmentation used in this thesis.

Image Augmentation

Image augmentation is a very important part of data preprocessing. It helps in artificially enlarging the dataset, making the network more robust against some changes in the quality or saturation of the image and improving its performance. It is also used to help the detection work if the sun is shining or the illumination is very low due to a cloudy sky or different times of the day. Also, a problem in the dataset for this thesis is the missing variation, since all of the images originate from the same camera and angles. Augmentation is also a possibility to decrease the overfitting of the model.

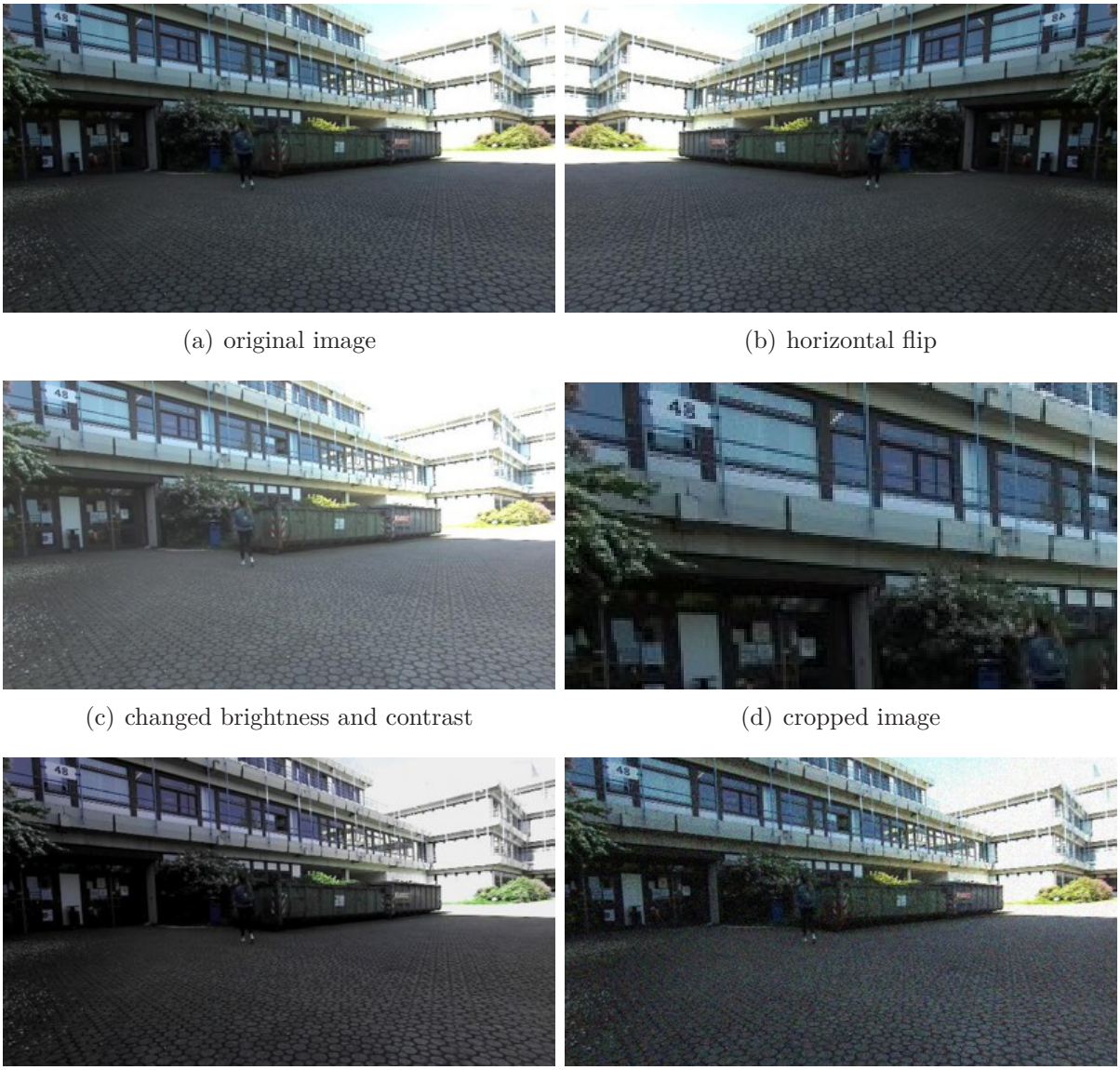
The amount of images in the dataset is quite low after the selection of suitable images therefore augmentation provides a viable option to amplify the performance of the network. There are 8 of the options chosen, that the TensorFlow object detection API provides. This also means, that the augmentation is not made manually before the training, but automatically during the training. The chosen options include:

1. horizontal flip shown in figure 4.6(b) and
2. brightness adjustment shown in figure 4.6(c) together with
3. contrast adjustment and
4. hue adjustment shown in figure 4.6(e) and
5. saturation adjustment and
6. image crop shown in figure 4.6(d) and
7. colour distortion and
8. gaussian patches shown in figure 4.6(f)

These augmentations are chosen because of the task the model should be able to solve. This means that a flip of the image in another way than horizontal would not make much sense, since the camera on the robot does usually not give pictures to the model that are upside down. Brightness is important because of the weather. It can be a problem if the objects get illuminated too much or too little. The changing of the brightness as well as hue, contrast, and saturation are an attempt at countermeasure. Colour distortion can happen for the most diverse reasons and it is better to prepare the model for this. The crop and gaussian patches augmentation are there to take the camera and the possible problems with it into consideration, as well as help the detection of partial objects.

4.1.3 Dataset Overview

After the preprocessing, the dataset consists of 13.466 annotated images. But a very uneven distribution of annotated instances per class can be seen. This is foreseeable because some classes naturally have more objects than others. And some can be seen from many points of the route and thus also have more instances. The number of building 11 is seen the least by the robot with only 60 instances. To even this, class-balancing is utilized on the 9 classes having less than 1000 instances. This means that some of the images, with objects of these classes, get duplicated. Therefore images are used, where only this class has an object. After this, a better result is imaginable, because of the more even distribution helping the network to learn all classes sufficiently.



(a) original image

(b) horizontal flip

(c) changed brightness and contrast

(d) cropped image

(e) changed hue

(f) added gaussian noice

Figure 4.6: Examples of the image augmentations used

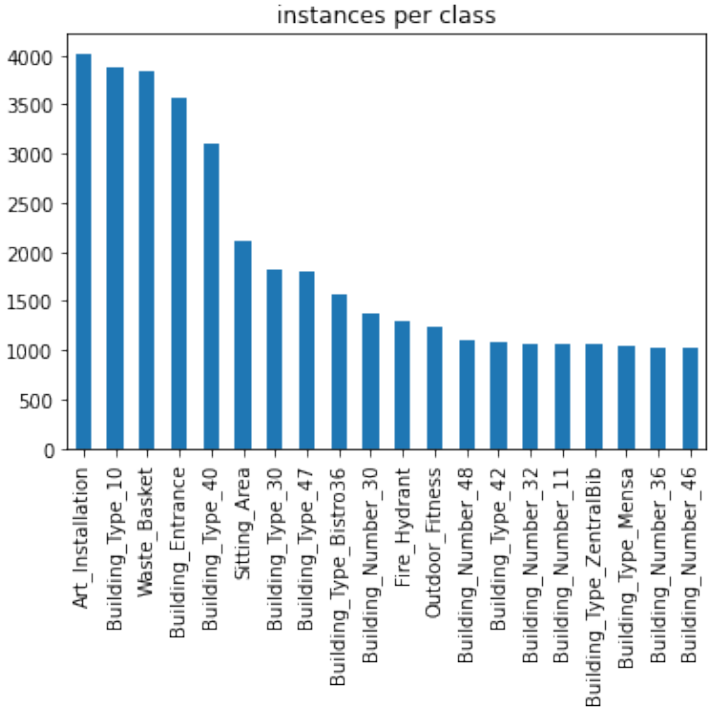
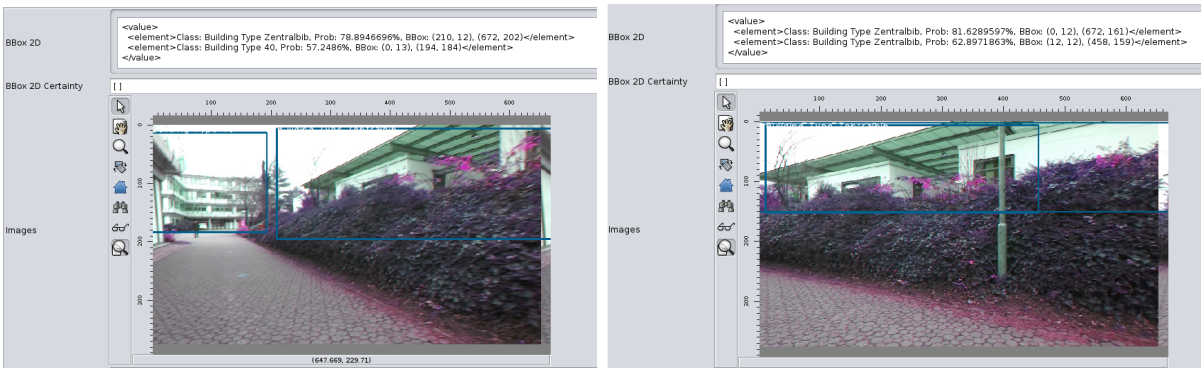


Figure 4.7: Instances per class

4.1.3.1 Dataset 1

The first dataset consists of 18.868 images of which 13.466 images are unique and 5.402 are duplicated images. All of them are labeled and the class distribution can be seen in figure 4.7. Every class has more than 1000 instances. Of course, there are still classes that have more instances than others, but these are mostly classes, that are also objects in OSM and therefore especially important to detect. Artworks (Art_Installation), building entrances (Building_Entrance), and waste bins (Waste_Basket) are the objects that are found very often on the OSM map. As well as benches (Sitting_Area) and hydrants (Fire_Hydrant), but these have a naturally lower instance count since there are not that many hydrants on the route chosen for this thesis for example.

This version of the dataset proves to be insufficient in the terms of false-positive detections. Although correct predictions are made, if there is an object to be detected in the image, as can be seen in figure 4.8(a). The model struggles, if there is no object to be detected. And in figure 4.8(b) can be seen, that predictions are made nonetheless. This is of course undesired behavior for the model and the solution to this problem is the insertion of images in the dataset, where no object that should be detected can be seen and then train the model further. A solution for this problem is proposed in the second version of the dataset explained in the next section.



(a) One true positive and one false positive detection

(b) Two false positive detections

Figure 4.8

	Images	Annotated Images	Copied Images	Images without annotation
Dataset after Preprocessing	13466	13466		
Dataset 1	18868	13466	5402	
Dataset 2	19858	13466	5402	990

Figure 4.9: Comparison of the different datasets with total images and the annotated, copied and not annotated portions

4.1.3.2 Dataset 2

To solve the problem of the first dataset, 990 additional images without annotated objects are inserted. The second dataset consists of 19.858 images of which 13.466 are unique, 5.402 are duplicated and 990 are not annotated, which makes up 5% of the total number of images used in the dataset. With this addition, the model is more robust in case of a phase in the drive, where no landmark can be detected.

In figure 4.9 an overview of the datasets and their image numbers is given for more clarity. Dataset 2 is the dataset the model is trained with and the changes that are made in each dataset can be seen in the number of images. Of course, data augmentation is not taken into account at this time, because it happening simultaneously with the training.

After the data is preprocessed, which is usually the same for every network, a model needs to be chosen to provide the architecture of the network.

4.2 Model

A detection architecture that unites speed, accuracy, and memory into a full functioning model is often sought. There are possibilities to achieve some of it, but a method must always be composed of different building blocks according to the requirements of the task. The TensorFlow object detection API has a few options that should be considered for this thesis.

method	top-1 err.	top-5 err.
VGG [40] (ILSVRC'14)	-	8.43†
GoogLeNet [43] (ILSVRC'14)	-	7.89
VGG [40] (v5)	24.4	7.1
PReLU-net [12]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Figure 4.10: Error rates (%) of single-model results on the ImageNet validation set [He 16]

Meta Architecture

At first, a meta-architecture has to be chosen. There is a choice of almost every modern meta-architecture like SSD, Faster R-CNN, and SSDlite in the TensorFlow object detection API model zoo, where a collection of the pre-trained model is maintained. The outputs of the models differ as some are boxes and some masks and the models differ in the respectively used feature extractor.

Feature Extractor

The feature extractor is first applied to the image and ensures the obtaining of high-level features [Huang 17]. For the model, the choice of a feature extractor needs to be carefully considered as it is responsible for the memory, speed, and performance of the resulting detector [Huang 17]. This is because of the number of parameters and types of layers directly influenced by the feature extractor. Feature extractors that can be chosen with the TensorFlow object detection API are Visual Geometry Group (VGG), ResNet, Inception. MobileNet as well as Inception ResNet.

Number of Region Proposals

For some of the pre-trained models the number of region proposals sent to the box classifier at test time can be adjusted to individual needs [Huang 17]. If computation needs to be saved fewer boxes can be sent, but this may result in reducing the recall [Huang 17].

In the next section the selection of the model for this thesis is discussed, as well as a in-depth overview over the chosen model.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 4.11: Architecture of Resnet with the building blocks shown in brackets and numbers of blocks stacked. Downsampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2. [He 16]

4.2.1 Model Selection

The model used for this thesis should have some properties like:

1. accuracy should be high
2. memory usage should not be too high
3. inference time should not be too high

To achieve this the TensorFlow object detection API proposes different solutions, but for this thesis, the meta-architecture Faster R-CNN and the feature extractor ResNet50 are chosen. This is a proven combination and with the implementation used by the TensorFlow object detection API, which is explained in the next section, all the targets can be reached.

4.2.2 Model Overview

Now the feature extractor is explained since Faster R-CNN is already touched on in 2.5. ResNet50 is a 50 layer residual net, that has a smaller error rate than VGG as can be seen in figure 4.10. The architecture of ResNet50 consists of a 3-layer bottleneck block resulting in 50 layers as shown in figure 4.11. But the implementation used by the TensorFlow object detection API and also in this thesis makes a few changes to this architecture to change the output stride to 16 instead of 32 to increase the mean average precision. This is achieved by changing the conv5_1 layer to have stride 1 instead of 2 with the reduced stride being compensated by atrous convolutions in further layers [Huang 17].

After the data preprocessing is made and a suitable model is found and understood, the actual training can begin. Therefore a few model depending steps have to be taken, which is why the model selection should be made before the actual training preparation.

Class ID	Class Name
1	Building_Entrance
2	Fire_Hydrant
3	Waste_Basket
4	Outdoor_Fitness
5	Building_Number_48
6	Building_Type_40
7	Building_Type_30
8	Building_Type_Bistro36
9	Art_Installation
10	Building_Type_ZentralBib
11	Sitting_Area
12	Building_Type_Mensa
13	Building_Number_30
14	Building_Type_42
15	Building_Type_47
16	Building_Number_36
17	Building_Number_46
18	Building_Number_32
19	Building_Type_10
20	Building_Number_11

Figure 4.12: Objects that can be detected with their class id and class name

4.3 Training

To train the Faster R-CNN model with the TensorFlow object detection API a few steps have to be taken first. Because the images are annotated in 4.1.2 in the PASCAL VOC format, they can instantly be converted to tf_record files. This file format stores data in a sequenced structure and is optimized to use with TensorFlow.

Before that, a separation into training and validation sets has to be made, with 90% of the images in the training set and 10% of the images in the validation set. An additional test set is utilized later in the experiment section. In this section further images are generated from a test video and these are used to evaluate the model.

After this conversion a train.record and a val.record file are existent, which can be used for training and validating the model. There is also the need for a label.pbtxt file, where all class names are stored together with their ID, which denotes the class number in figure 4.12. Also, the .config file has to be adjusted for the own prerequisites and dataset. After all these preparations the training can be started. For this thesis, the network is trained for 250.000 steps and reaches a loss below 0.1.

After the training ends the question, of how good the resulting model is, stays open. The loss can give only information about the model training on the training data but not about the final performance and the performance on unknown data. To investigate these, the next chapter presents some techniques to rate the model and also to test the model in different areas.

5. Experiments

The experiments part of this thesis deals with the evaluation of the model resulting from the training. For this, a few considerations must be made.

At first, a general check of how well the model performs on data that is very similar to the data from the training is interesting for an initial overview. This is handled by various evaluation metrics provided by the TensorFlow object detection API. Two well-known protocols for evaluation are the PASCAL VOC protocol and the COCO protocol. The COCO dataset is already presented in 4.1 and the PASCAL VOC dataset is touched on in the same section. These two datasets have their own metrics, which can be used for other datasets and help establish general evaluation metrics to efficiently compare models to other models.

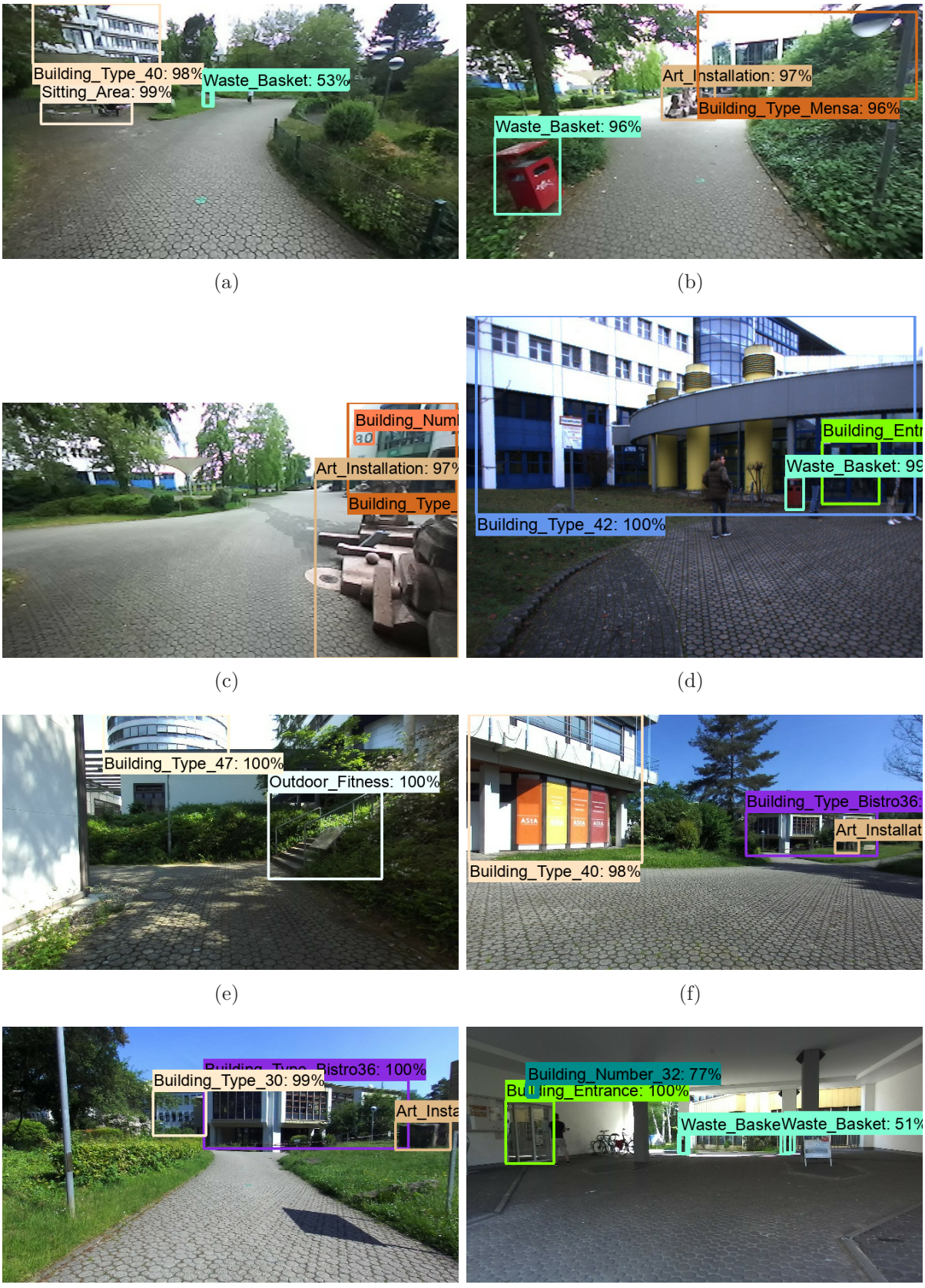
Then an evaluation of the test video is made, first by evaluating it visually and finding problem fields and then analyzing a small subset of it with the two metrics mentioned before. This is done to not only have a visual impressions but also to be able to quantify these and asses performance of the model on unseen data.

The last experiment moves away from the sole evaluation of the model and explores the applicability of the model in the localization of the robot. For this, the implementation of the work in [Kunz 21] is used. The goal is to show, that the developed detector can be used with the localization algorithm. At this time it is not explored how well the localization works and how well the matching with landmarks from the OSM database works.

The proposed experiments are further elaborated in the next sections.

5.1 Evaluation of the model

To get an idea of how well the final model behaves in the wild, evaluation metrics are important measures. They generalize the observations, which can be made by just looking at the results. There is no consensus on the metrics used in different projects, but the



(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figure 5.1: Object detection on example images with true predictions part 1

protocols of different object detection challenges can be used to achieve the comparability of different models. These are also suitable for the evaluation of the model on its own since there are some differences in the output of each protocol that give valuable information about the model. But before this, a look at the predictions of the model can give a first impression on the capabilities and problem fields, which then should be investigated further.

(a) (b)

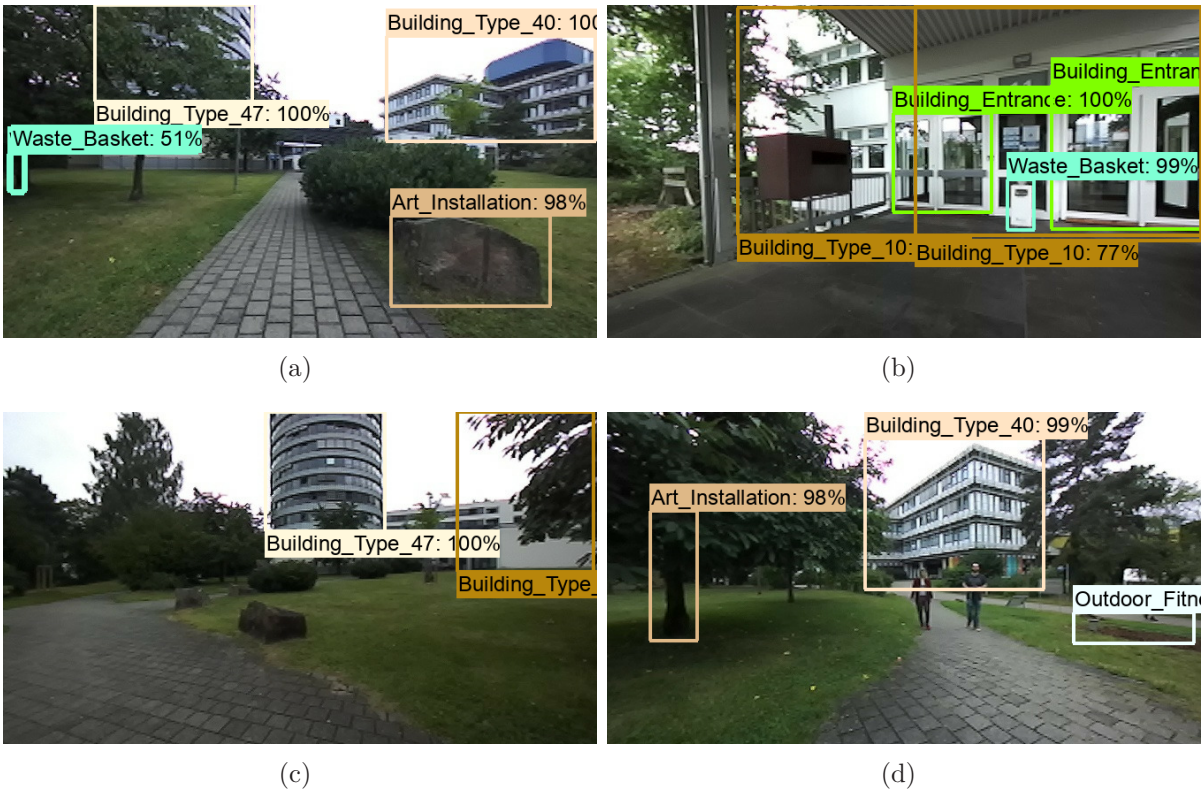
(c) (d)

(e) (f)

Figure 5.2: Object detection on example images with true predictions part 2

As the images in figure 5.1 and figure 5.2 show, predictions on the validation set seem to be quite right. Many objects get detected and have high confidence. These are mostly big objects, like the buildings and objects in the foreground. That gives the impression that the prediction on big objects might be good. But smaller objects and objects that

are reasonably far away get detected as well.



(a) (b)

(c) (d)

Figure 5.3: Object detection on example images with false predictions part 1

To discuss a few problem fields figure 5.3 and figure 5.4 show detections that are not right or objects that should be detected but are not. There seems to be an issue with the detection of the building numbers as can be seen in figure 5.3(b), figure 5.4(d), figure 5.4(e), figure 5.4(f) and figure 5.4(h). Also the false detection of buildings with the number range of 10 seems to be a problem as shown in figure 5.3(c), figure 5.4(a), figure 5.4(b) and figure 5.4(c). In figure 5.4(g) both of these classes make problems. Also in figure 5.3(a) a rock is detected as artwork, but this is not a rock that belongs to the class artwork. And in figure 5.3(d) the model detects a tree as artwork.

These issues are foreseeable because the buildings of type 10 are rather generic-looking buildings with white walls and no color on the windowsill like buildings with type 30 or timber frames like buildings with type 40. There are other white buildings, mostly in the vicinity of building 47, which are not in the number range 10. The building numbers are problematic because the dataset already lacks many good examples for them. While other classes are easier to collect the numbers can often be seen in only a short time frame and then they are often quite far away. Some of the objects, like the number 46 and the number 48 are really small signs as can be seen in figure 5.5. It is even for humans hard to decide if it is a 6 or an 8 because they are so similar. Other numbers are a bit easier

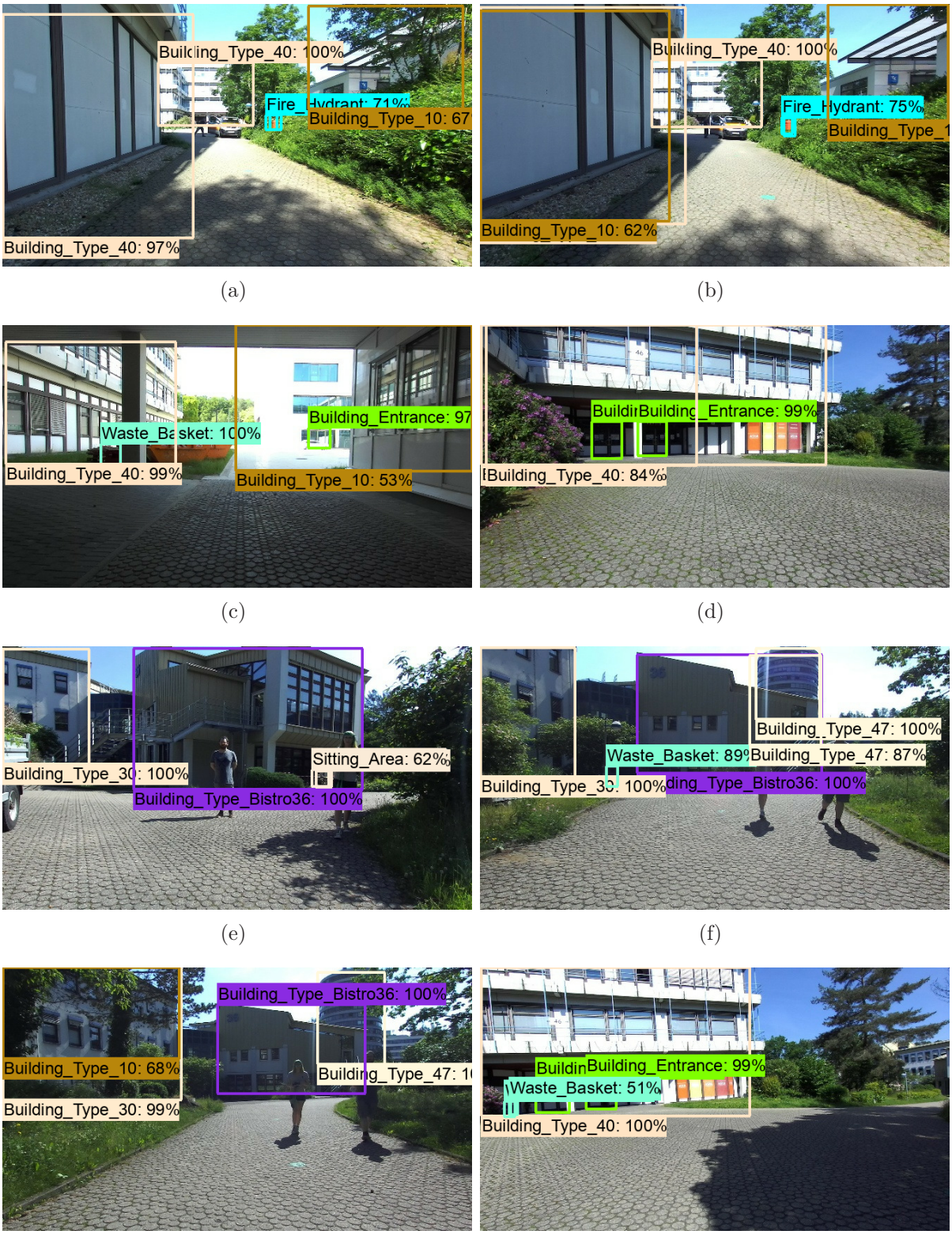


Figure 5.4: Object detection on example images with false predictions part 2

and therefore get detected better.

(a) (b)

Figure 5.5: Building number 46 and building number 46

A more accurate analysis of these first impressions is provided by the metrics in the next sections.

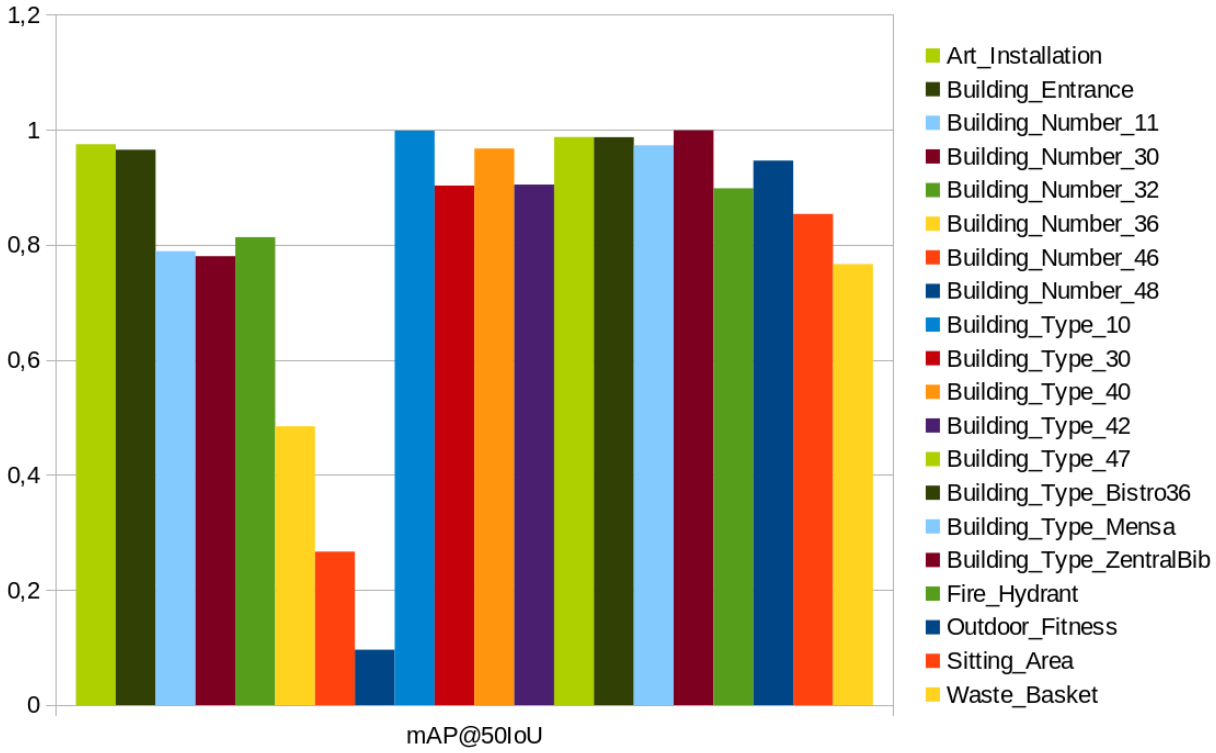


Figure 5.6: Mean Average Precision of different objects

5.1.1 PASCAL VOC evaluation protocol

The PASCAL VOC evaluation protocol was originally developed as an evaluation tool for the PASCAL VOC challenge. With time it also proves to be a valuable tool to evaluate models not taking part in this challenge. Especially the mean average precision (mAP) metric implemented by PASCAL VOC is widely seen as a standard metric to evaluate models. This mAP score is the mean of the average precision over all classes using a single IoU threshold of 0.5, which means that a prediction is only counted as positive if the IoU is greater than 0.5 and that if there are more than one prediction of a single object only the one with the highest IoU score is counted as positive and the others as negative [Everingham 10].

PASCAL VOC does not only give the mean average precision but also an overview of the precision every class achieves. This is interesting for this thesis, because it is important to know, which objects can be reliably detected and which not. In the figure 5.6 the previously made assumptions about the problem fields of the detection are strengthened. The classes of the building numbers 36, 46, and 48 have the lowest scores, with the class of number 46 having a really low score. This explains the inadequate detection of the class and also of the other two. Every other class has acceptable scores and this is also confirmed by the visual evaluation. Of course, the scores for the other building numbers are low in comparison to the other classes but still high enough to give usable predictions, since many of them are detected correctly.

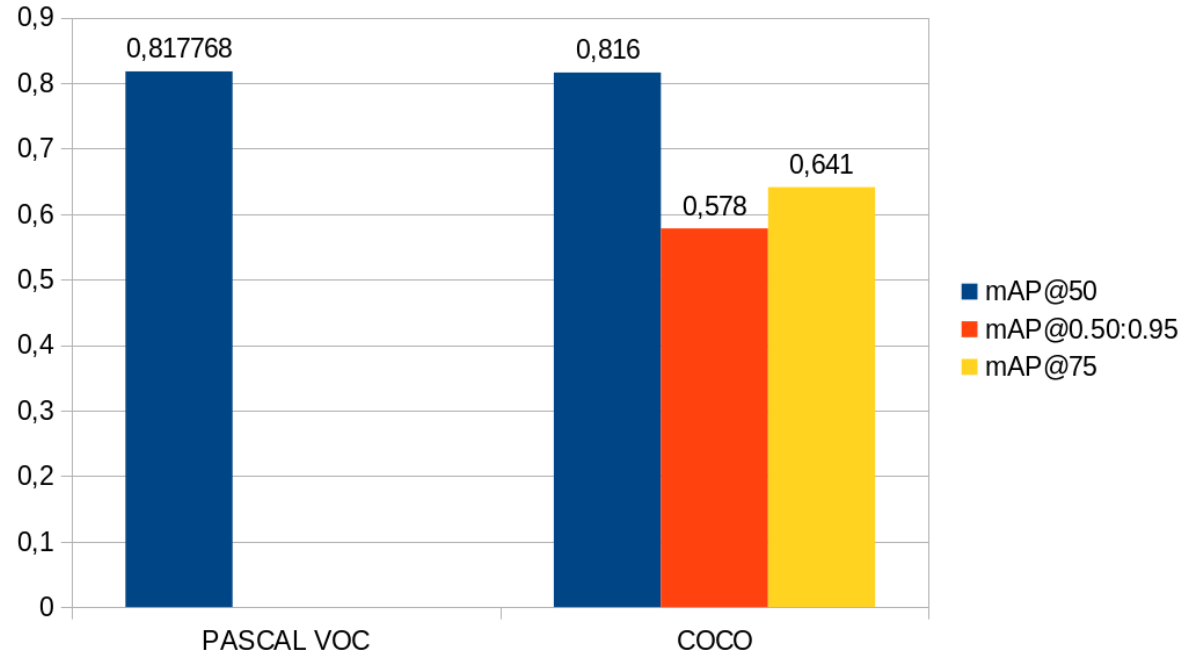


Figure 5.7: Mean Average Precision of the model on the validation set

The mAP of the model denotes the performance of its detection and in figure 5.7 it can be seen, that the performance of the proposed model on the validation set is very good with a value of 0.817768. This is further explored in the next section with the COCO protocol, which provides other scores to consider.

5.1.2 COCO evaluation protocol

The COCO evaluation protocol was first developed as an evaluation metric for the COCO challenge but provides now different metrics than the PASCAL VOC protocol and is also very important for the evaluation of object detection models. In difference to the PASCAL VOC protocol, the COCO protocol provides not only the mAP computed at the IoU value of 0.5 but also a version where the mAP is averaged over multiple IoU thresholds. In the mAP@0.5:0.95 score, there are 10 thresholds to be considered [Coco21]. For this first, the average precision is calculated for each class for an IoU threshold of 0.5. This is then repeated for the threshold values at every 0.05 increase. At the end, all of these values are averaged to calculate the mAP@0.5:0.95 score.

In figure 5.7 the different outcomes of these calculations can be seen. The mAP@0.5:0.95 score of the model is lower than the two other scores. This means that the model may not be that good at the localization of the detected objects, since this is rewarded with the mAP@0.5:0.95 score. But this could also be because of the difficult annotation of some of the objects, where no clear borders could be observed. The building type class is complicated, because of the buildings that are merging into each other but belong to the same class. This happens at the meeting point of building 48 and building 46 as well as at

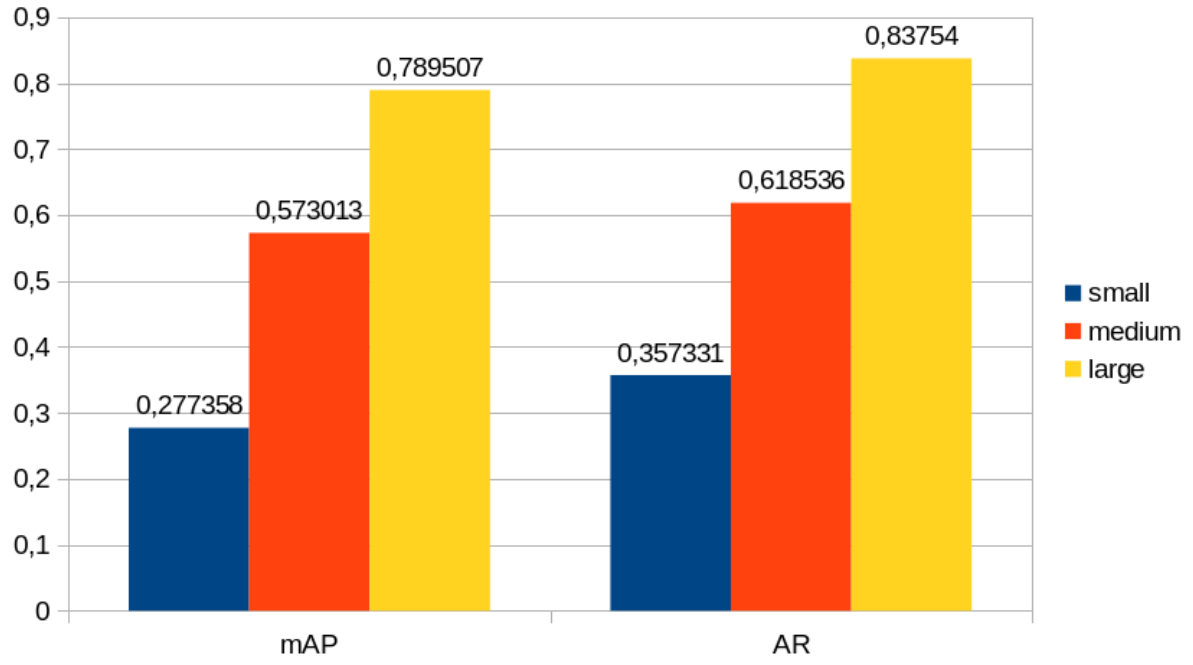


Figure 5.8: Comparison of mean average precision and average recall of different sized objects

the meeting point of building 36, 34, and 32.

More important for this thesis is the consideration of different object sizes in the precision metric. This is valuable information because the visual evaluation recognizes a problem field of the model in this thesis concerning small objects. In figure 5.8 the mAP and average recall of different sized objects is presented. Here "small" denotes an area covering less than 32^2 on pixel scale, "medium" denotes an area covering more 32^2 on pixel scale and less than 96^2 on pixel scale and "large" denotes an area covering more than 96^2 on pixel scale [Coco21]. The mAP scores of the different area sizes show, that the visual evaluation is right about the problem field of small objects. With an mAP of 0.277358, these areas are not well detected by the model. But because of the usage of the model as a localization help, the objects that are far away and thus smaller, are not that interesting. The detection should be good for an area directly in front of the robot and therefore usually contains large or medium-sized objects. For these, the detection is usable with scores of 0.573013 for medium-sized objects and 0.789507 for large objects.

Also taken into consideration in figure 5.8 is the recall score of the model, which shows, that a big portion of the large objects gets detected by the model. This is important because of the task the model is used for. If not many instances of objects would be detected, the model would not be usable, because it is important to detect as many of the instances of the objects as possible. The recall scores are higher than the precision scores, which means that the model does not produce that many false-negative predictions. This is important for the task of localization because a false negative prediction could

negatively influence the position adjustment of the robot.

Overall the model seems to work well on the validation data and seems suitable for the task of supporting the localization. But the question of how good the performance of the model is on data that does not belong to the original dataset remains. To answer it, a test video is made of a drive on the campus. This video is taken by the same robot as the dataset but considerably later in the year in November 2021. The evaluation of this test data is discussed in the next section.

5.2 Evaluation on a Test Dataset

To get an idea of the generalization capability of the model as well as the performance on data, which is taken from a real-life scenario a test video of a drive on the campus is made. The route driven this time covers only a few of the available ones. And of some objects, there are very few instances. To even this a bit further, images of objects are added. These depict mostly interesting classes, which are hard to detect. The evaluation of this test dataset should give a general idea of the usability of the model.

5.2.1 Visual Evaluation

At first, a visual evaluation of the detections of the model is made to gauge the abilities of the model. In figure 5.9 a selection of detections is shown. The problem fields discussed in section 5.1 are still there as can be seen in figure 5.9(a) and figure 5.9(g) where a building in the vicinity of building 47 is falsely detected as a building of type 10. This is still a hard class, because of the white building. Other images, like figure 5.9(b), 5.9(c), 5.9(e) and 5.9(f) show true detections. In figure 5.9(d) it can be seen that a particular object of the class outdoor fitness is hard to detect, if the weather is too cloudy and thus makes the image rather dark. This is also a difficult object because it is just a small line in the image, which makes it difficult to detect. In figure 5.9(f) the object is successfully detected, but the illumination of the images is better and the angle of view makes it easier to see the object. In figure 5.9(h) another example of how the angle influences the success of detection can be seen. The artwork and building of type 40 are detected, but the central library building is partly covered by other objects in the image and shown in a bad angle for detection since it is mostly seen from the other side, where it has a better identifiable facade.

The model is tested on Finroc using the whole test video. There the observation can be made, that most of the objects get successfully detected, though sometimes a few frames are needed. This is because the objects are not clearly visible or too blurry, because of the movement of the vehicle. In the figure 5.10 an example for the wrong detection of an object can be seen in figure 5.10(a), which is corrected in the next frame as can be seen in figure 5.10(b). The building number does not get detected, which is because of the problems of the size of the number and the brightness of the image, which are already discussed above. Otherwise, the detection has the same problem fields already discussed above. But the successive frames of a video make it possible to correct the detections because the same scene is visible for some time and a correct prediction of all the objects

Figure 5.9: Object detection on test images with true and false predictions

(a) (b)

Figure 5.10: Example detection on the test video

is usually made in at least one of them.

To analyse the test dataset further, the same protocols as used in the previous section are now utilised.

5.2.2 PASCAL VOC evaluation protocol

Again the PASCAL VOC protocol is used to determine the mAP of the detections for each class. In this case, not every possible class is in the test set, because of the route of the test drive.

In figure 5.11 a comparison of the mAP scores for the different classes can be seen. It should be taken into account, that the test set used to calculate these scores is rather small with only 204 images and therefore a low count of most objects. But the scores also strengthen the assumption that the detection of the building numbers is not good, especially in the case of building number 46. It can be assumed that the score for the number 48 would not be much better, because of the problem with the size of the number on the building. But the scores for the building types are all rather high. This is good for the localization of the robot because they are visible in more frames due to their size whereas numbers, at least numbers 46 and 48, are only visible for a very short time. The artwork also has a high mAP score, which is good, because these artworks are visible all over the campus and also important for localization. This also applies to the waste bins, entrances, and fire hydrants. Two classes that have a high mAP score on the validation set, but a rather low one on the test set are the fitness equipment and the benches. The problem with the angle and illumination of the fitness equipment class is already discussed in section 5.2.1. But for the benches, the test set has just a few instances, since on the route only one bench object is visible. And this is a rather hard one to detect because it is partly concealed by nature elements.

In figure 5.12 it can be seen, that the mAP of the model on the test set is lower than on the validation set. But an mAP of 0.518 is still acceptable and it is only a portion of the

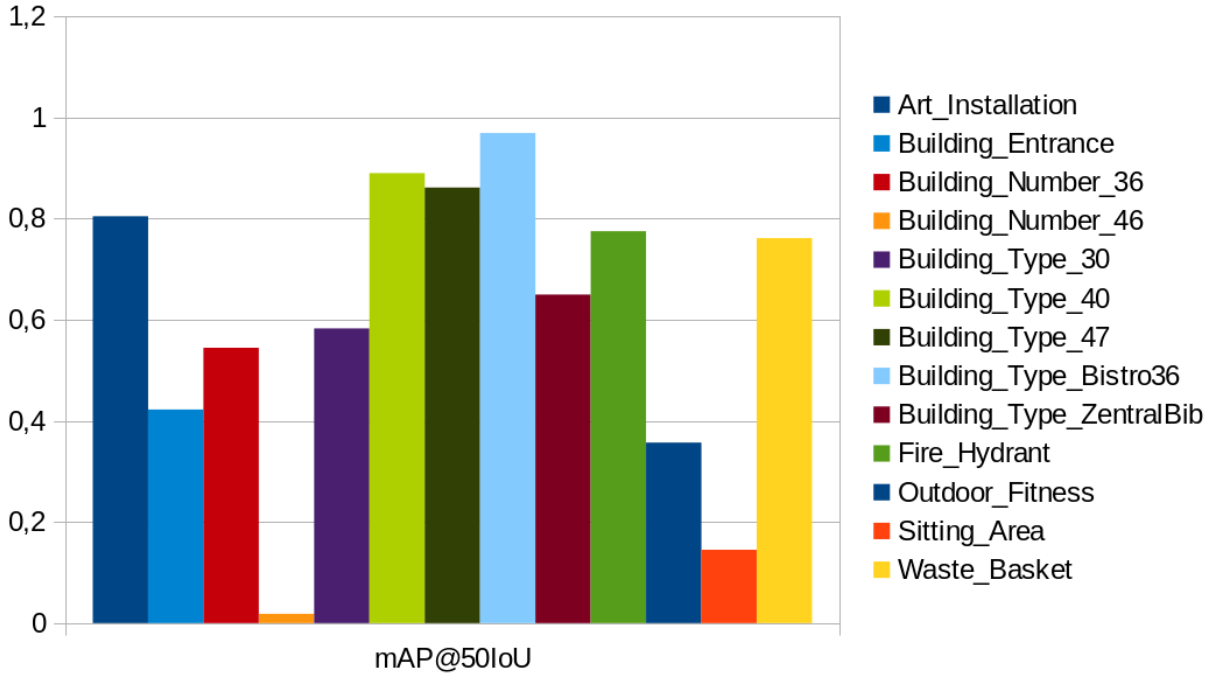


Figure 5.11: Mean Average Precision of different objects

objects so it only denotes the direction the precision takes. Further exploration of these scores is made in the next section with the COCO protocol, which again provides different scores to consider.

5.2.3 COCO evaluation protocol

As in the evaluation of the validation set, the validation on the test set yields a lower score on the mAP@0.50:0.95 metric. As discussed above, this means that the localization of the model may be not the best. This only means that the bounding boxes made by the model deviate from the ones of the ground truth. This is not that important for the task of the model, since the exact bounding boxes are not necessarily needed, but only a standardized version of them.

More important is the comparison of the detection of different-sized objects. This is again achieved by using the metrics provided by the COCO protocol, which gives the detection performance on objects of different sizes. In figure 5.13 the objects covering a small area of under 32^2 on pixel scale have a low mAP score. This is expected because of the results on the validation set. Small objects do not have a good detection, which explains the bad mAP scores of the building numbers in contrast to the scores of the building types and other objects. The objects covering a medium area of more than 32^2 on a pixel scale and less than 96^2 on a pixel scale also have a rather low mAP score and the objects covering an area greater than 96^2 on a pixel scale have the best mAP of all of them. It has to be kept in mind, that this is only an excerpt of the possible objects.

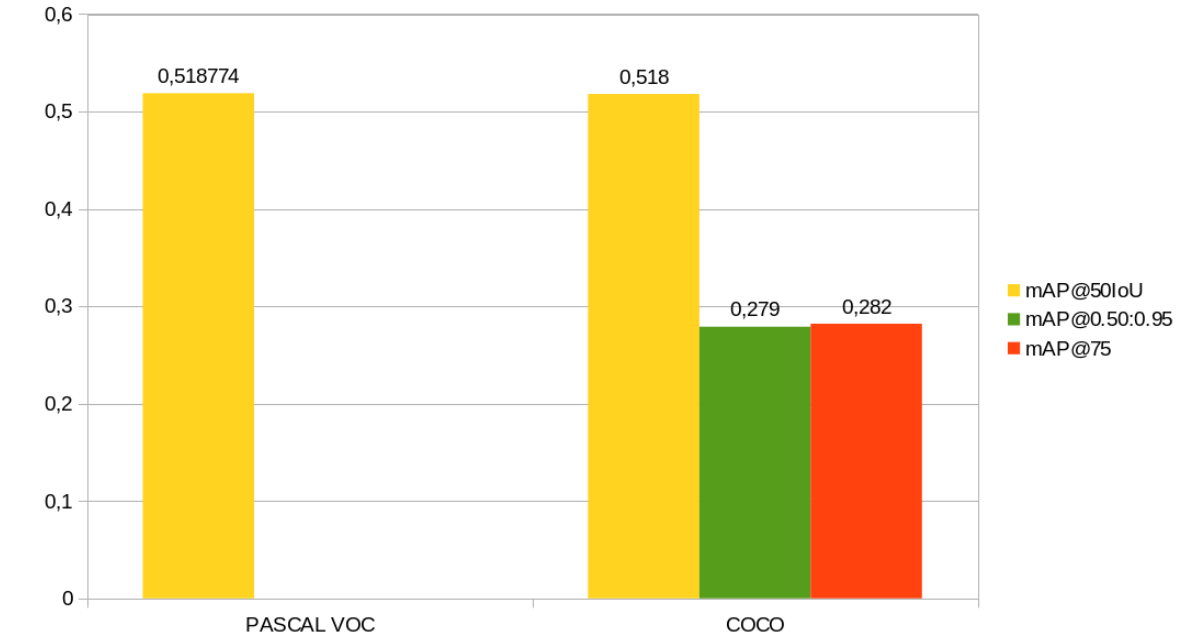


Figure 5.12: Mean Average Precision of the model on the test set

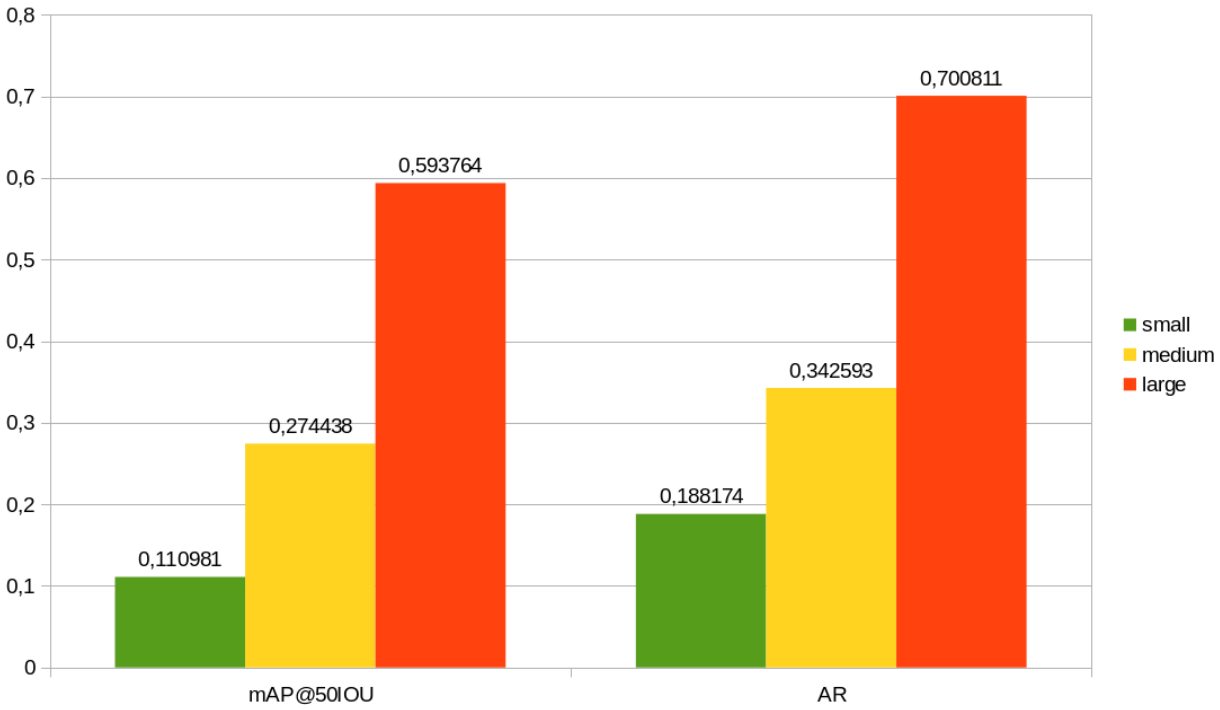


Figure 5.13: Comparison of mean average precision and average recall of different sized objects

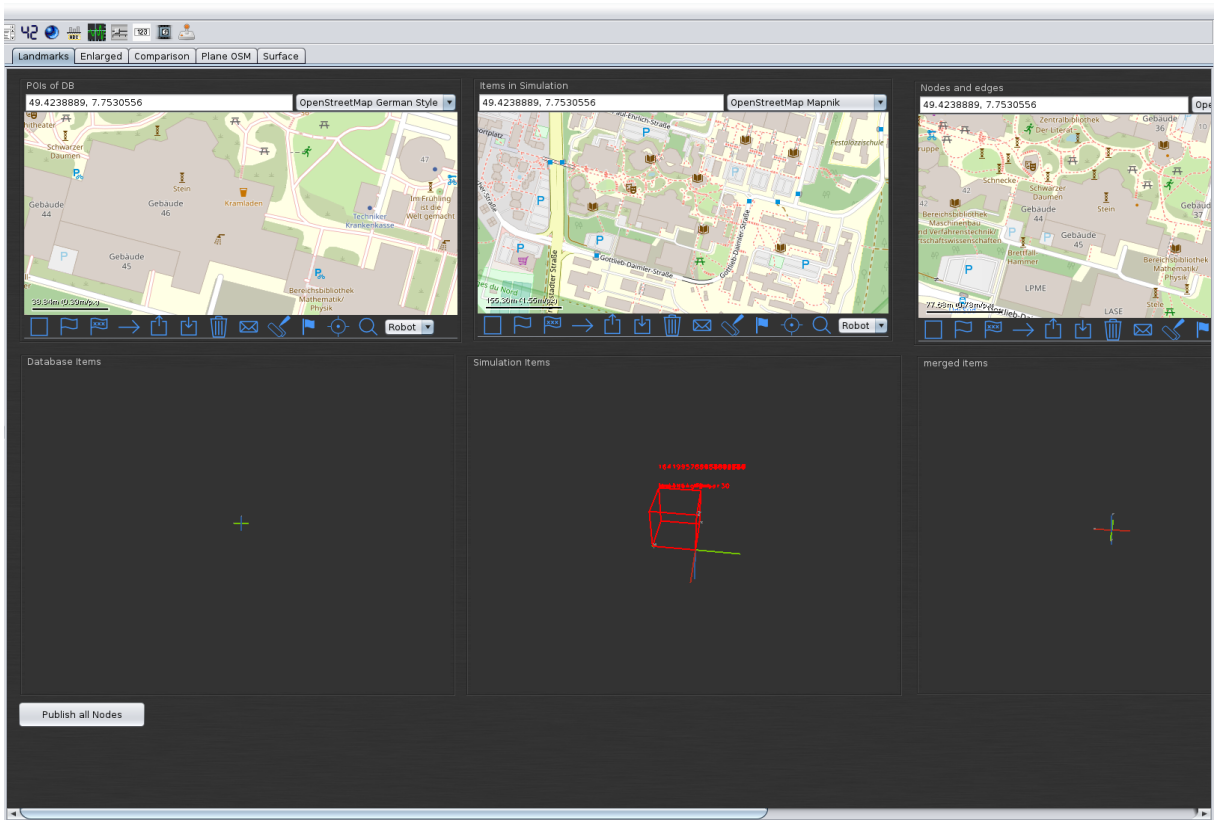


Figure 5.14: Test of the model on the osm_localisation finroc library developed in [Kunz 21]

In figure 5.13 the recall scores for the different sized objects are also depicted. This is also an important measure for the model since it is not only interesting if the detected objects are true detections but also how many of the objects, which are there, are detected. The model in this thesis has a high recall score on objects covering a large area. This makes it suitable for the task of detection for localization because if only a few of the objects are not detected, the model is not good, but here most of them are detected.

In the next section, it is tested, if the model developed in this thesis can be used to aid the localization of the algorithm proposed in [Kunz 21].

5.3 Testing the model with the Localisation Algorithm

The localization algorithm provided in [Kunz 21] should use a detector to provide the bounding boxes and tags to compare to the ones in the database to aid the localization of the robot. A possible detector is developed in this thesis and how it should be tested if this detector can be used with the algorithm. At this time it is not tested how well the localization works or how well the data provided by the model can be matched to the data in the database.

For this experiment, a video of a drive on the campus is used to simulate the robot driving on the campus. The model is then used to detect objects in this video with the finroc campus_detection library. This library is already used to test the model on finroc and visualize the output, which consists of the class that is detected, as well as its confidence score and the bounding box values. At this time the 3D point cloud-based detection is not used to get a local position, since this is an optional parameter not needed to show that the model can be used with the algorithm.

To test the applicability of the model the output of the campus_detection library is given to the finroc osm_localisation library. There a visualization of the output is made of the bounding box with the respective label of the class as can be seen in figure 5.14. The detected objects could now be compared to objects in the database to find a match for the position of the robot.

6. Conclusion

In this chapter, a summary of the work in this thesis is made. Therefore a review of the previous chapters and an outlook on the usability and problem fields of the model are given. As well as a few ideas of how this model, made to aid the localization of the robot, can be improved. Also, an outlook on the integration in the in [Kunz 21] implemented localization algorithm is given together with the proposal of additions to integrate the model better in this work.

6.1 Summary

This thesis aims to develop an object detector, able to detect the OSM landmarks, to aid the Delivery-Robot of the RRLAB to localize itself on the campus. For this purpose, a detector is considered with its architecture and the basic classes for a dataset to train the network. Then the dataset is created by driving two routes on the campus, which are popular and generate enough data to test if this proposed detector is a viable option to aid the localization. After this, the network is trained with the data gained from the preprocessing, which includes a selection of images suitable for the task as well as class-balancing to even the unbalanced distribution of instances for some classes. Simultaneously with the training, a further attempt at making the model more robust is made with image augmentation. This also generates a bigger dataset to train on, which is needed to prevent the overfitting of the model.

The Experiment chapter of this thesis aims to evaluate the previously trained model to gain an insight into the applicability. It is discovered, that the model can detect objects with a large area on the pixel scale efficiently on the validation and the test set. A problem field is discovered as well, which explains, why some classes perform poorly. The model struggles to detect small objects, which mostly concern the numbers of the buildings. Also, a problem field is the brightness and reflection of objects triggered by the sun. As well as insufficient illumination of some objects, which makes it difficult for the model to detect them against the background. The model is also tested for its applicability with the localization algorithm proposed in [Kunz 21] and it is shown, that the model can theoretically be used to aid the algorithm in the localization of the robot.

6.2 Outlook on future work

There are two areas in which the work of this thesis can be improved and continued in the future. One area is the further investigation of the model together with the in [Kunz 21] proposed localization algorithm. For this, the 2D detections of this thesis should be converted to 3D detections based on the point cloud and camera frustum to get the local position, angle, and distance of the object. This can then be used to answer the question of how well the data provided by the model can be used to match the data in the database and thus also how well the localization works. It could also be investigated, how well the detection of the buildings can be used to estimate the GNSS signal strength in advance since the signal strength is often lacking in the vicinity of these.

The second area is the further improvement of the model itself. At this time on two routes on the campus are covered, but besides these, there are a few other possible routes that can be driven by the robot. These would also provide a few additional landmarks like bus stops and other artwork and buildings, not yet included in the dataset. As well as more variation for some of the objects like fire hydrants and different angles for buildings. Another imaginable improvement could be achieved by driving the same routes already used in this thesis at different times of the day and in different seasons. This would make the model more robust because in the winter the sun sets earlier than in the summer and this could be a problem for the detection of some of the objects, which merge with the background too easily.

Bibliography

[Abadi 16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard et al, "Tensorflow: A system for large-scale machine learning", in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 2016, pp. 265–283.

[Accuracy 20] "Classification: Accuracy", `https://developers.google.com/machine-learning/crash-course/classification/accuracy`, 2020.

[Aggarwal 18] C. C. Aggarwal, *Neural Networks and Deep Learning*, Springer, Cham, 2018.

[Albawi 17] S. Albawi, T. A. Mohammed, S. Al-Zawi, "Understanding of a convolutional neural network", in *2017 International Conference on Engineering and Technology (ICET)*. 2017.

[Alpaydin 14] E. Alpaydin, *Introduction to Machine Learning*, MIT Press, 2014.

[Chipka 18] J. B. Chipka, M. Campbell, "Autonomous urban localization and navigation with limited information", in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE. 2018, pp. 806–813.

[Coco21] "Detection Evaluation", Git code.

[Everingham 10] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, A. Zisserman, "The pascal visual object classes (voc) challenge", *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[Forsyth 11] D. Forsyth, J. Ponce, *Computer Vision: A Modern Approach. (Second edition)*, Prentice Hall, nov. 2011.

[Girshick 15] R. Girshick, "Fast r-cnn", in *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[Goodfellow 16] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016. `http://www.deeplearningbook.org`.

[He 16] K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[Huang 17] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama et al, "Speed/accuracy trade-offs for modern convolutional object detectors", in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7310–7311.

[Jabbar 15] H. Jabbar, R. Z. Khan, "Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)", *Computer Science, Communication and Instrumentation Devices*, pp. 163–172, 2015.

[Kunz 21] M. Kunz, A. Vierling, P. Wolf, K. Berns, "Localization using OSM landmarks", in *2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE)*. 2021, pp. 228–233.

[Lin 14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, "Microsoft coco: Common objects in context", in *European conference on computer vision*, Springer. 2014, pp. 740–755.

[Liu 16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, "Ssd: Single shot multibox detector", in *European conference on computer vision*, Springer. 2016, pp. 21–37.

[Mitchell 97] T. M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

[Murphy 22] K. P. Murphy, *Probabilistic Machine Learning: An introduction*, MIT Press, 2022.

[Nilwong 19] S. Nilwong, D. Hossain, S.-i. Kaneko, G. Capi, "Deep learning-based landmark detection for mobile robot outdoor localization", *Machines*, vol. 7, no. 2, p.25̃, 2019.

[O'Shea 15] K. O'Shea, R. Nash, "An Introduction to Convolutional Neural Networks", *CoRR*, vol. abs/1511.08458, 2015.

[PostBot 21] "DeliveryRobot of the Technical University Kaiserslautern", `https://agrosy.informatik.uni-kl.de/roboter/postbot`, 2021.

[Precision 20] "Classification: Precision and Recall", `https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall`, 2020.

[Redmon 16] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You only look once: Unified, real-time object detection", in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[Ren 15] S. Ren, K. He, R. Girshick, J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks", *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[True/False Positive/Negative 20] "Classification: True vs. False and Positive vs. Negative", `https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative`, 2020.

[Tzutalin] Tzutalin, “LabelImg”, Git code.

[ZED 21] “ZED 2 - AI stereo camera”, `https://www.stereolabs.com/zed-2/`, 2021.

[Zhang 21] A. Zhang, Z. C. Lipton, M. Li, A. J. Smola, “Dive into Deep Learning”, *arXiv preprint arXiv:2106.11342*, 2021.

[Zou 19] Z. Zou, Z. Shi, Y. Guo, J. Ye, “Object Detection in 20 Years: A Survey”, 2019.